Stan Krute

Grafica e suoni con il Commodore 64

Il piacere del computer

Il piacere del computer

1 *Tom Rugg e Phil Feldman* 32 programmi con il PET
2 *Rich Didday* Intervista sul personal computer, hardware
3 *Tom Rugg e Phil Feldman* 32 programmi con l'Apple
4 *Ken Knecht* Microsoft Basic
5 *Paul M. Chirlian* Pascal
6 *Tom Rugg e Phil Feldman* 32 programmi con il TRS-80
7 *Rich Didday* Intervista sul personal computer, software
8 *Herbert D. Peckham* Imparate il Basic con il PET/CBM
9 *Karl Townsend e Merl Miller* Il personal computer come professione
10 *Karen Billings e David Moursund* Te ne intendi di computer?
11 *Thomas Dwyer e Margot Critchfield* Il Basic e il personal computer, uno: introduzione
12 *Don Inman e Kurt Inman* Imparate il linguaggio dell'Apple
13 *Thomas Dwyer e Margot Critchfield* Il Basic e il personal computer, due: applicazioni
14 *Luigi Pierro* Il manuale del CP/M
15 *Carlo Sintini* A scuola con il PET/CBM
16 *David Johnson-Davies* Il manuale dell'Atom
17 *David E. Schultz* Il libro del Commodore VIC 20
18 *Jim Huffman e Robert Bruce* Il "debug" nei personal computer
19 *John M. Nevison* Programmazione in Basic per l'uomo d'affari
20 *Mark Harrison* Imparate il Basic con lo ZX81
21 *Ronald W. Anderson* Dal Basic al Pascal
22 *Herbert D. Peckham* Imparate il Basic con il Texas TI 99/44
23 *Sergio Borsani* A scuola con il Texas TI 99/4A
24 *Jerry Willis e Deborrah Willis* Come usare il Commodore 64
25 *Mark Harrison* Imparate il Basic con lo Spectrum
26 *Carlo Sintini e Costantino Mustacchio* A scuola con il Commodore 64
27 *David A. Lien* Imparate il Basic con l'IBM PC
28 *Ken Tracton* Introduzione al Lisp
29 *Fabio Mavaracchio* Programmi in Basic per l'elettronica
30 *Ian Stewart e Robin Jones* Il linguaggio macchina dello Spectrum
31 *Tom Rugg, Phil Feldman e Clarence S. Wilson* 32 programmi per il VIC 20
32 *Merl Miller e Mary A. Myers* Introduzione all'Apple Macintosh
33 *Stam Krute* Grafica e suoni con il Commodore 64
34 *Jerry Willis e William Manning* Come usare l'IBM PCjr
35 *Tom Rugg e Phil Feldmann* 32 programmi con il Commodore 64

Stan Krute

Grafica e suoni con il Commodore 64

franco muzzio & c. editore

Titolo originale *Commodore 64 graphics and sound programming*
Traduzione di Andrea Panagia

Prima edizione: gennaio 1985
ISBN 88-7021-273-4

Indice

APPENDICI

Prefazione

Confesso che i calcolatori mi piacciono veramente, sono strumenti molto divertenti, ma allo stato attuale ci sono due miglioramenti che mi piacerebbe fossero realizzati: un abbassamento dei prezzi e una maggiore qualità della grafica e del suono.

Gli amici della Commodore stanno spingendo l'industria nella giusta direzione. Hanno realizzato il meraviglioso VIC-20 solo tre anni fa e adesso è arrivato il Commodore 64 che offre una grafica potente e buone capacità sonore, nonché una memoria notevolmente più capace, un hardware molto più flessibile e un prezzo che fa venire l'ulcera alla concorrenza.

Ho passato molto del mio tempo insegnando alla gente l'uso dei calcolatori e mi sono accorto, specialmente lavorando con i ragazzi, che tutti volevano imparare a disegnare e a suonare con il calcolatore e, poiché la cosa mi interessava particolarmente, ho sempre incoraggiato questa inclinazione. Dal momento in cui ho avuto le prime notizie sul Commodore 64, ho subito pensato che volevo imparare al più presto tutti i suoi trucchi per poi comunicarli agli altri.

Per i primi otto mesi ho avuto il piacere di esplorare il suono e la grafica del Commodore 64 e con questo libro desidero mettervi al corrente delle mie scoperte, combinando il calcolatore con arte, musica, logica e giochi. Spero che vi incoraggi a continuare per questa strada. Il Commodore 64 è un veicolo particolarmente versatile per questo tipo di impieghi: usatelo bene e mettete in comune con gli altri i vostri risultati.

Introduzione

Questo libro è stato scritto per chi ha già esperienza con i calcolatori e per programmatori di medio livello che vogliano introdursi all'uso delle possibilità grafiche e sonore del Commodore 64. Il libro copre una vasta area delle possibilità della macchina in questi due interessanti campi. I 68 programmi che contiene sono scritti in Basic in modo che risultino il più possibile chiari e leggibili.
L'unico libro reperibile che si occupi degli stessi argomenti è il *Commodore 64 Programmer's Reference Guide*. È un ottimo libro, che probabilmente vorrete inserire nella vostra biblioteca se vi appassionerete a questi argomenti, ma il suo unico difetto è che forse è un po' troppo avanzato per la maggior parte delle persone che si avvicinano per la prima volta al calcolatore. Quando avrete finito di leggere il volume che vi trovate tra le mani sarete in grado di passare al manuale Commodore senza un interprete pagato.
Avete bisogno di un Commodore 64, di un televisore di buona qualità e di un dispositivo che vi consenta di immagazzinare i programmi che vi presenteremo in questo libro. Se volete migliorare l'aspetto grafico vi conviene munirvi di un monitor di buona qualità, ad esempio il monitor a colori della Commodore. Per immagazzinare i programmi potete utilmente impiegare il registratore a cassette della Commodore, ma se prenderete più sul serio la vostra attività di programmatore l'unità disco è un lusso che vorrete concedervi. L'unità disco Commodore 1541 è molto solida e allo stesso tempo economica anche se ha la tendenza a surriscaldarsi nel corso di sedute di lavoro particolarmente lunghe (oltre le 12 ore).
I primi sei capitoli di questo libro si occupano della grafica. Imparerete cosa sono gli *sprite*, la grafica a caratteri e a mappa di bit. Il Commodore 64 rende questo genere di programmazione estremamente più semplice

di ogni altra macchina di questo tipo attualmente in commercio. Potrete ottenere notevoli risultati grafici con semplici programmi Basic grazie al potente hardware contenuto nel Commodore 64.
I successivi tre capitoli trattano la generazione del suono con il Commodore 64. Il circuito integrato che genera il suono equivale a un sintetizzatore a tre voci di buona qualità. Ancora una volta con il semplice ausilio del Basic sarete in grado di ottenere risultati che richiederebbero dei complessi programmi in linguaggio di assemblatore su altri calcolatori simili. Infine il capitolo 10 vi insegnerà ad unire grafica e suono.
Penso che il miglior modo di imparare a programmare sia seguendo degli esempi e provando. Per questo abbiamo inserito nel libro 63 programmi, oltre la metà dei quali viene discussa estesamente nel corso del testo. Ogni capitolo termina con un breve riassunto e una serie di esercizi studiati in modo da chiarire i punti fondamentali trattati nel capitolo. Sono anche inclusi 30 programmi da realizzare con le relative soluzioni, in modo che possiate familiarizzarvi col calcolatore.
Alcune parti possono risultare poco chiare al primo impatto: per questo abbiamo cercato di inserire il maggior numero di illustrazioni che insieme alle appendici potessero chiarire il significato del testo. So infatti per esperienza quanto sia fastidioso leggere un libro che attira la nostra attenzione ma che frappone alla nostra comprensione una foresta impenetrabile di termini geregali. Ho anche incluso speciali tabelle di codifica che potrete copiare e utilizzare per realizzare i vostri sprite e i vostri caratteri grafici.
Sempre tenendo conto del fatto che i programmi sono scritti in Basic, ho cercato di renderli il più possibile chiari e modulari. Ho passato buona parte del mio tempo programmando in Pascal e in assembler e ho cercato di realizzare gli esempi seguendo la disciplina di programmazione così appresa.
Pur non essendo un linguaggio molto potente, il Basic è semplice e di facile apprendimento per chi si avvicina per la prima volta al calcolatore e può essere proficuamente usato se lavorate con cura.
Ho cercato di evitare di usare negli esempi quello stile di programmazione che chiamo "spaghetti scotti", cioè quel tipo di codice dove in ogni riga si incontrano GOTO, incomprensibili nomi di variabili, enunciati senza alcuna relazione tra loro e così via. Sicuramente un codice di questo tipo avrà prestazioni decisamente scarse, e se volete programmi di veloce esecuzione sviluppate un algoritmo migliore o trascrivete parti del vostro programma in linguaggio macchina.
A questo punto penso che i preliminari siano sufficienti, potete iniziare la vostra avventura.

COME USARE QUESTO LIBRO

Se non avete molta confidenza con il Commodore 64, vi converrà fare un po' di pratica. Leggete i primi capitoli della *Commodore 64 User's Guide*, che viene fornita insieme al computer; costituiscono una buona introduzione alle operazioni fondamentali della vostra macchina. Se avete poca esperienza di programmazione in Basic leggete un buon manuale introduttivo scelto tra i molti reperibili in commercio. Tornate a questo libro quando avrete acquistato maggior dimestichezza con la macchina e con il linguaggio.
Questo libro è stato redatto in modo da consentire un apprendimento graduale e attivo; si tratta di un utile strumento per chi inizia ad esplorare le capacità della macchina, è completo di spiegazioni dei vari problemi comunemente incontrati, di esempi di programmazione e di esercizi, ed è corredato da utile materiale di riferimento.
I dieci capitoli sono strutturalmente simili; ognuno tratta una mezza dozzina di problemi. Una breve introduzione presenta il problema in esame, quindi viene riportato un breve programma da provare sul vostro calcolatore, che viene successivamente discusso. Vengono anche suggerite alcune interessanti modifiche da apportare al programma originale. Alla fine del capitolo si trovano alcuni brevi esercizi di ricapitolazione e di programmazione. Sono comunque riportate le risposte ai quesiti proposti e le possibili soluzioni agli esercizi.
Usando l'ordine postale allegato in fondo al libro potete acquistare un dischetto contenente i programmi riportati nel corso del volume, il che vi eviterà di doverli riscrivere; dovrete solo caricarli e farli partire.
In ogni caso se preferite potete riscriverli voi stessi: si tratta di un procedimento molto semplice e gli unici problemi che potrete incontrare sono relativi ai caratteri iconici.
Il Commodore 64 infatti dispone di un insieme di caratteri di controllo che consentono di controllare la posizione dello schermo in cui vengono riportate le informazioni desiderate e i movimenti del cursore, di pulire lo schermo, cambiare il colore dei caratteri o di invertirlo. Potete compiere queste operazioni direttamente dalla tastiera come riportato nelle pagine 14-17 del manuale del Commodore 64, ma è senza dubbio più interessante farlo dall'interno di un programma.
Come? È sufficiente definire una costante di tipo stringa che contiene il carattere di controllo. È sufficiente racchiuderlo tra apici sia in un assegnamento che in un comando PRINT. In questo modo una volta che la stringa richiamata viene inviata sullo schermo funzionerà come un comando da tastiera.
L'unico problema si presenta quando dobbiamo stampare o scrivere un

ICONE DI COLORE

Icona	Tasti da premere	Che cosa fa	Icona	Tasti da premere	Che cosa fa
	CTRL-1	Colore del testo nero		C= – 1	Colore del testo arancione
	CTRL-2	Colore del testo bianco		C= – 2	Colore del testo marrone
	CTRL-3	Colore del testo rosso		C= – 3	Colore del testo rosso chiaro
	CTRL-4	Colore del testo cyan		C= – 4	Colore del testo grigio scuro
	CTRL-5	Colore del testo violetto		C= – 5	Colore del testo grigio medio
	CTRL-6	Colore del testo verde		C= – 6	Colore del testo verde chiaro
	CTRL-7	Colore del testo blu		C= – 7	Colore del testo blu chiaro
	CTRL-8	Colore del testo giallo		C= – 8	Colore del testo grigio chiaro

ALTRE ICONE

Icona	Tasti da premere	Che cosa fa	Icona	Tasti da premere	Che cosa fa
	CLR/home	"Home" del cursore		Shift–CLR/home	Pulische lo schermo
	CRSR ↕	Cursore giù		Shift–CRSR ↕	Cursore su
	CRSR ⇄	Cursore a destra		Shift–CRSR ⇄	Cursore a sinistra
	CTRL–9	Reverse attivato		CTRL-0	Reverse disattivato

FIG. I-1. Icone di colore del Commodore 64.

programma che contiene questo tipo di caratteri che vengono stampati come caratteri grafici. Ad esempio, il comando che ripulisce lo schermo viene rappresentato da un cuore capovolto e il movimento a sinistra del cursore da una riga verticale in colore inverso.

Questi sono i cosiddetti caratteri iconici. Quando vedete uno di questi caratteri in un listato di un programma, dovete capire quale comando rappresenta e quale tasto bisogna premere per ottenerne l'esecuzione. La tabella nella figura 1.1 dovrebbe chiarire ogni dubbio, infatti vi sono riportati tutti i caratteri iconici usati in questo libro, i tasti da usare per ottenerli e i comandi che rappresentano. Se incontrate un assegnamento o un'istruzione di stampa che contengono strani caratteri tra apici, fate riferimento a questa tabella.

Un buon consiglio per chi preferisce trascrivere da sé i programmi proposti è di salvare ogni programma prima di provarlo, in modo che se un errore fa saltare il sistema non sarete costretti a riscrivere tutto.
Se il programma che avete battuto non funziona, avete probabilmente commesso un errore nella trascrizione. Esaminate accuratamente riga per riga ciò che avete scritto e quindi usate il meraviglioso editor del Commodore 64 per correggere gli errori. Se a questo punto il programma non funziona ancora, confrontatelo di nuovo riga per riga con l'originale e se non trovate nulla di sbagliato ribattete le righe dove sono segnalati gli errori, anche se sembrano giuste; può darsi che in questo modo il tutto funzioni.
Una volta che il programma funziona, sia che lo abbiate caricato da disco o da nastro sia che lo abbiate ribattuto, guardate quello che succede e provate a far riferimento a quello che avete scritto. Provate a immaginare l'effetto di ogni singola riga del vostro programma sui suoni e le figure che vedete sullo schermo, quindi ritornate a leggere la discussione del programma su questo libro.
A questo punto arriva il divertimento: provate a modificare il programma già provato, caricatelo di nuovo e osservate cosa succede quando lo fate partire.
Fate qualche altra modifica, cambiate il codice di qualche colore, spostate qualche figura, fate partire di nuovo il programma; questo è uno dei metodi migliori per imparare l'essenza della tecnica grafica e sonora.
Se volete veramente diventare esperti programmatori nel campo della grafica e del suono dovrete dedicare a questa attività buona parte del vostro tempo. Pensate a una immagine o a un suono interessante e cercate di scrivere un programma che li generi partendo da zero. Portate la macchina ai suoi limiti e cercate di superarli.
Cominciate a scrivere alcuni programmi più lunghi che facciano uso di una maggiore varietà di suoni e immagini generati con tecniche diverse, provate a fare quello che viene generalmente ritenuto impossibile, impiegate parte del vostro tempo leggendo qualsiasi programma che riuscite a reperire che si occupi di questi argomenti, cercando di capire perché l'autore ha usato una certa tecnica e provate a sviluppare un algoritmo migliore. Leggete le appendici di questo libro e quelle di tutti i libri di questo genere, leggete gli articoli delle riviste che vi possono interessare. Tenetevi aggiornati, cercando di sviluppare idee vostre partendo da quelle degli altri.
Ancora un paio di suggerimenti: il Commodore 64 inizia ogni sessione di lavoro con i caratteri rappresentati in blu su fondo blu in modo da risultare quasi illeggibili, cambiateli quindi a bianco su nero premendo **CTRL-2** e battendo questi due comandi:

```
POKE 53280,12
POKE 53281,0
```

Inoltre, se state usando una unità disco potrete incontrare delle difficoltà: leggete sul manuale operativo della vostra unità le informazioni sul programma DOS Wedge e usatelo ogni volta che iniziate una seduta di lavoro; questo vi consentirà di usare comandi più potenti, versatili e allo stesso tempo più semplici.

CAPITOLO UNO

Un primo sguardo agli sprite

Questo capitolo vi introduce a una delle più potenti caratteristiche del Commodore 64: gli *sprite*. Imparerete come creare uno sprite e come muoverlo sullo schermo. Imparerete anche come cambiare la forma e il colore del vostro sprite.

CHE COS'È UNO SPRITE?

Accendete il vostro televisore. Avvicinatevi allo schermo: vedrete dei piccoli punti luminosi. Le immagini televisive sono composte da migliaia di piccoli punti di questo tipo.
Il più piccolo punto che un calcolatore può accendere sullo schermo di un televisore viene chiamato *pixel*, che in inglese è un'abbreviazione per *picture cell*, cella elementare d'immagine. Uno sprite è un tracciato di pixel che il vostro Commodore 64 può muovere sullo schermo.
Lo sprite di base è composto da 24 pixel di larghezza per 21 pixel di altezza. Date un'occhiata alla figura 1.1. Se moltiplicate 21 righe per 24 colonne troverete un totale di 504 pixel con cui potete divertirvi. Se non ci credete provate a contare i riquadri.
In uno sprite potete fare in modo che un particolare pixel sia visibile o invisibile. Potete vedere un esempio di questo nella figura 1.2. Potete creare innumerevoli figure differenti usando questi 504 pixel, circa 2.207. 107. 920. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000. 000.

FIG. 1.1. Il tracciato di base per gli sprite del C-64 copre 504 pixel.

000. 000. 000. 000. 000. 000, uno spazio sufficientemente vasto per la vostra creatività.

COME DEFINIRE IL TRACCIATO DI UNO SPRITE

Avete bisogno di un modo per dire al calcolatore quali pixel di uno sprite devono essere visibili e quali no. Potete farlo con una codifica numerica per gruppi di otto pixel.
Date un'occhiata alla figura 1.3. Ognuno degli otto riquadri rappresenta un pixel e ha un numero che lo contraddistingue e che rappresenta il valore del pixel. Per esempio, il primo da sinistra ha un valore pari a 128, mentre quello più a destra vale 1 e così via.
Adesso osservate la figura 1.4. Alcuni dei riquadri sono stati riempiti.

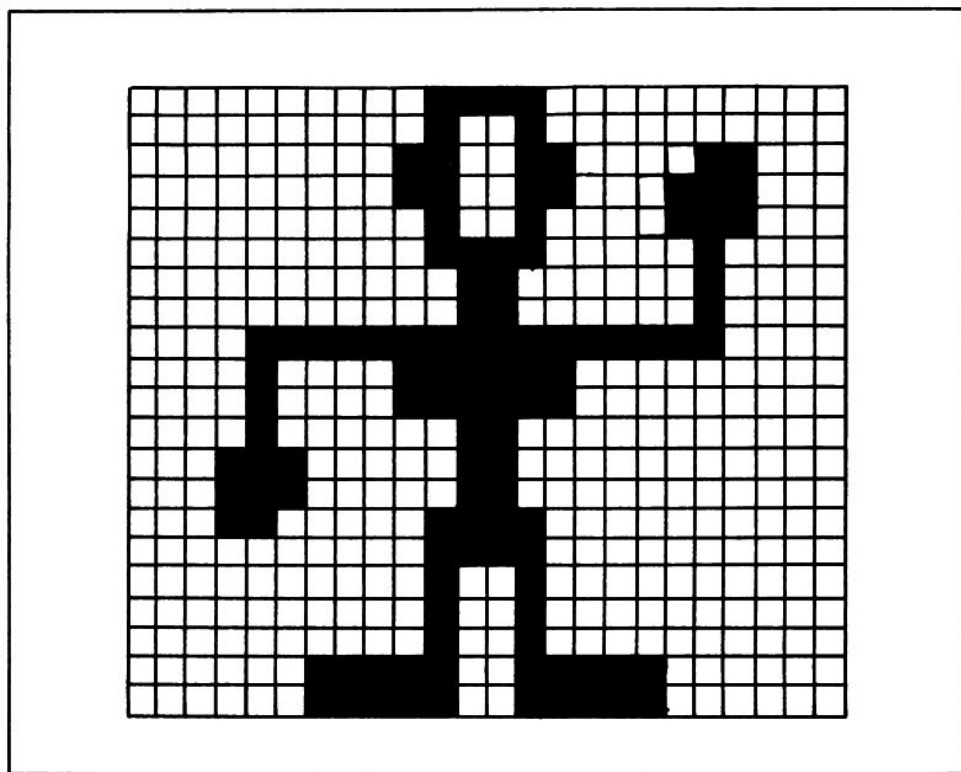



FIG. 1.2. Un'immagine realizzata rendendo visibili alcuni pixel in uno sprite.

128	64	32	16	8	4	2	1

FIG. 1.3. Valori usati per codificare gruppi di otto pixel.

Se sommate i valori che corrispondono ai riquadri pieni avrete 85: infatti 64 + 16 + 4 + 1 = 85. La figura 1.5 mostra altri esempi di come si possono codificare dei pixel con questo metodo.
Esaminate ora la particolare forma di codifica utilizzata per gli sprite e rappresentata nella figura 1.6. La tabella è composta da 21 righe per 24 colonne come richiesto. Ogni riga è divisa in tre parti per la codifica numerica e ognuna di queste parti è a sua volta suddivisa in otto

FIG. 1.4. Codifica di un gruppo di otto pixel.

Schema:

Codice numero:

128 64 32 16 8 4 2 1

128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255

128 64 32 16 8 4 2 1

32 + 16 + 8 + 4 = 60

128 64 32 16 8 4 2 1

128 + 16 + 8 + 1 = 153

128 64 32 16 8 4 2 1

(niente) = 0

FIG. 1.5. Altri esempi di codifica.

colonne. In testa a ogni colonna è segnato il valore di codifica relativo. Ogni riga può venire rappresentata con tre codici numerici, uno per ogni gruppo di otto colonne appartenenti alla stessa riga. Poiché vi sono 21 righe, otterrete 63 codici numerici. Questi codici vanno poi inseriti nel Commodore 64 nel giusto ordine: da destra a sinistra in ogni riga e dalla prima all'ultima riga, dall'alto verso il basso.

Colonna numero	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	Numero codici		
Valori	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1			
Riga 0																											
Riga 1																											
Riga 2																											
Riga 3																											
Riga 4																											
Riga 5																											
Riga 6																											
Riga 7																											
Riga 8																											
Riga 9																											
Riga 10																											
Riga 11																											
Riga 12																											
Riga 13																											
Riga 14																											
Riga 15																											
Riga 16																											
Riga 17																											

FIG. 1.6. Una speciale forma di codifica per gli sprite.

Per definire uno sprite vi consigliamo di seguire questi quattro punti:

1. Fate una copia della tabella di codifica.
2. Tracciate una figura riempiendo i riquadri che corrispondono ai pixel che volete accendere.
3. Calcolate i 63 codici numerici, uno per ogni gruppo di otto pixel.
4. Inserite nel calcolatore i codici nell'ordine richiesto.

La figura 1.7 mostra la codifica di uno sprite che rappresenta una piccola e amichevole creatura. Guardatela attentamente e cercate di capire come è stata realizzata la codifica. Rileggete le ultime pagine finché non avrete capito esattamente come funzionano le cose. Anche i più brillanti programmatori, usando le istruzioni più semplici, devono a volte rileggere più volte le cose poco chiare.
A questo punto potete cominciare a provare. Fotocopiate la pagina che contiene la tabella per la codifica e disegnate qualche sprite. Quindi calcolate i 63 codici che userete più avanti in questo capitolo; fate ora una pausa e continuate quando vi sarete riposati.

Colonna numero	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	Numero codici		
Valori	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1			
Riga 0																									0	60	0
Riga 1																									0	36	0
Riga 2																									0	102	24
Riga 3																									0	102	56
Riga 4																									0	36	56
Riga 5																									0	60	16
Riga 6																									0	24	16
Riga 7																									0	24	16
Riga 8																									15	255	240
Riga 9																									8	126	0
Riga 10																									8	126	0
Riga 11																									8	24	0
Riga 12																									28	24	0
Riga 13																									28	24	0
Riga 14																									24	60	0
Riga 15																									0	60	0
Riga 16																									0	36	0
Riga 17																									0	36	0
Riga 18																									0	36	0
Riga 19																									3	231	192
Riga 20																									3	231	192

FIG. 1.7. Esempio di codifica di uno sprite.

IL PRIMO PROGRAMMA PER GENERARE UNO SPRITE

Cominceremo con un semplice programma che disegna uno sprite. La figura 1.8 rappresenta a grandi linee la struttura del programma. La figura 1.9 contiene un listato del programma denominato "Un semplice sprite".

Osservate attentamente le figure. Inserite il programma nel vostro Commodore 64. Se non sapete come rappresentare i caratteri iconici impiegati fate riferimento al paragrafo "Come usare questo libro" contenuto nell'introduzione. Assicuratevi di aver salvato il vostro programma su disco o su cassetta una volta battuto. Quindi fatelo partire; schiacciate qualsiasi tasto per fermare l'esecuzione.

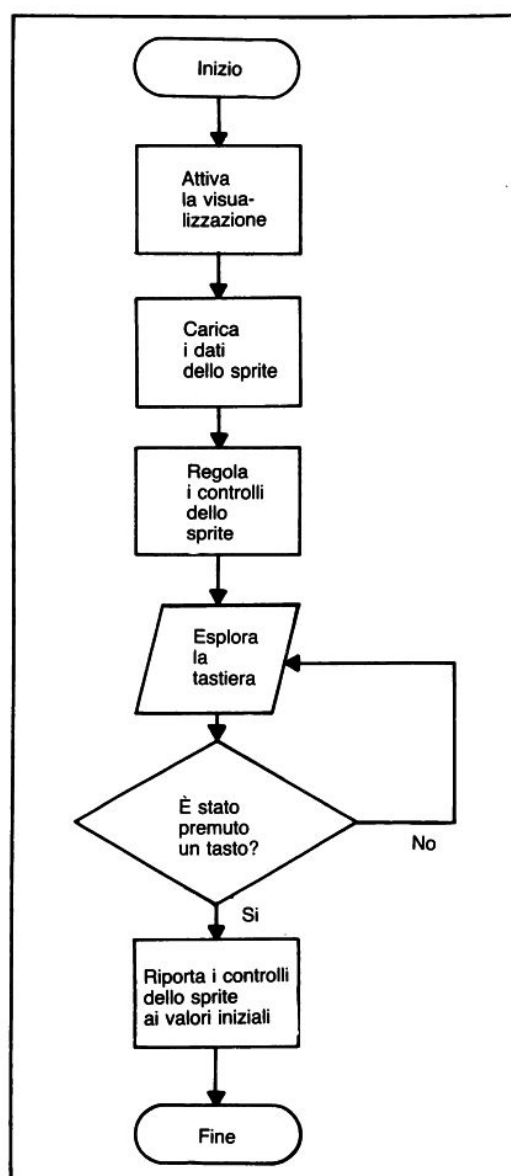



FIG. 1.8. Struttura del programma "Uno sprite semplice".

```
1000 REM *** UNO SPRITE SEMPLICE ***
1010 :
1020 :
1030 REM ** PROVVEDE UN FEEDBACK SULLO SCHERMO
1040 :
1050 PRINT "JQQQQQQQQQSTO PENSANDO"
1060 :
1070 :
1080 REM ** CARICA I DATI DELLO SPRITE
1090 :
1100 FOR N = 896 TO 958
1110 :    POKE N, 255
1120 NEXT N
1130 :
1140 :
1150 REM ** FISSA I CONTROLLI DELLO SPRITE
1160 :
1170 PRINT "J";       :REM PULISCE LO SCHERMO
1180 POKE 2040,14     :REM PUNTA AI DATI
1190 :
1200 VIC = 53248      :REM CHIP GRAFICO
1210 POKE VIC,170     :REM POSIZ. ORIZZONTALE
1220 POKE VIC+1,120   :REM POSIZ. VERTICALE
1230 POKE VIC+39,13   :REM COLORA DI VERDE
1240 POKE VIC+21,1    :REM ATTIVA LO SPRITE 0
1250 :
1260 :
1270 REM ** ASPETTA CHE VENGA PREMUTO UN TASTO
1280 :
1290 GET KP$
1300 IF KP$ = "" THEN 1290
1310 :
1320 :
1330 REM **        REINIZIALIZZA        **
1340 REM ** I CONTROLLI DELLO SPRITE **
1350 POKE VIC+21,0 :REM  INVERTE L'ORDINE
1360 POKE VIC+39,0 :REM  USATO PER INIZIALIZZARE
1370 POKE VIC+1,0  :REM  I CONTROLLI
1380 POKE VIC,0    :REM  DELLO SPRITE
1390 :
1400 END
```

FIG. 1.9. Listato del programma "Uno sprite semplice".

Il programma

Esaminate il listato. La prima sezione attiva è la linea 1050.

```
1050 PRINT "JQQQQQQQQQSTO PENSANDO"
```

Questa istruzione Basic ripulisce lo schermo e scrive un messaggio spostato verso il basso di alcune righe. Nulla è più fastidioso di un programma che non dà segni di vita mentre sta caricando i dati.
La seconda sezione del programma, tra le linee 1100 e 1120, carica i 63 codici dello sprite.

```
1100 FOR N = 896 TO 958
1110 :   POKE N, 255
1120 NEXT N
```

Per rendere questo programma il più comprensibile possibile, ho disegnato il più semplice sprite rappresentabile, quello in cui tutti i pixel sono accesi. In questo modo tutti i 63 codici sono rappresentati dallo stesso numero: 255. Il ciclo compiuto da queste istruzioni colloca i codici in 63 locazioni di memoria consecutive, i cui indirizzi vanno da 896 a 958.
La terza sezione del programma, compresa tra le linee 1170 e 1240, è la parte principale: guardate dapprima le righe tra 1170 e 1200:

```
1170 PRINT "♥";       :REM PULISCE LO SCHERMO
1180 POKE 2040,14     :REM PUNTA AI DATI
1190 :
1200 VIC = 53248      :REM CHIP GRAFICO
```

La linea 1170 ripulisce lo schermo, la 1180 dice al calcolatore in quali locazioni di memoria si trovano i dati relativi allo sprite. In che modo? Il vostro Commodore 64 può visualizzare contemporaneamente fino a otto sprite, numerati da 0 a 7. Quando gli dite di visualizzare lo sprite 0, la prima operazione che compie è andare a vedere il contenuto della locazione di memoria 2040 per trovare i valori di codifica dei pixel dello sprite 0. I numeri che si trovano qui vengono moltiplicati per 64. In questo caso moltiplicherà 14 per 64 e otterrà 896.
La linea 1200 assegna a una variabile denominataVIC il valore 53248. Che cos'è VIC?

Una digressione sul VIC-II

Il cuore delle capacità grafiche del Commodore 64 è un piccolo circuito integrato che viene ufficialmente chiamato 6567 Video Interface Chip, in breve VIC-II. (Il VIC-I era il 6560 usato nel VIC-20.) Questo dispositivo consente di visualizzare 25 righe per 40 colonne di caratteri o immagini con risoluzione di 320 per 200 pixel o otto sprite.
Trasmettendo alcuni particolari numeri in alcune locazioni di questo integrato è possibile controllarne alcune funzioni. Vi sono 47 locazioni indirizzabili nel VIC-II, che vengono anche chiamate registri. I registri del VIC-II iniziano dalla locazione 53248 e vanno fino alla 53294. Nell'appendice A sono contenute maggiori informazioni sui registri del VIC-II.

Torniamo al programma

Avendo assegnato alla variabile VIC il valore 53248 è possibile accedere alle locazioni di memoria successive semplicemente aggiungendo alla variabile il numero del registro al valore di VIC. Osservate ora le ultime quattro linee di questa parte:

```
1210 POKE VIC,170    :REM POSIZ. ORIZZONTALE
1220 POKE VIC+1,120  :REM POSIZ. VERTICALE
1230 POKE VIC+39,13  :REM COLORA DI VERDE
1240 POKE VIC+21,1   :REM ATTIVA LO SPRITE 0
```

Il registro 0 controlla la posizione orizzontale dello sprite 0. La linea 1210 del nostro programma gli assegna 170, all'incirca al centro dello schermo. Il registro 1 controlla la posizione verticale dello sprite 0 e nella linea 1220 gli viene assegnato il valore 120, che si trova nel centro verticale dello schermo. Il registro 39 assegna il colore con cui verranno visualizzati i pixel dello sprite 0. Il colore 13 corrisponde al verde chiaro. Leggete l'appendice F per avere una lista dei colori disponibili.
Una volta assegnati tutti questi valori, dobbiamo solo chiedere al VIC-II di mostrare sullo schermo lo sprite 0. La linea 1240 esegue quest'ultima istruzione. Il registro 21 viene usato per accendere o spegnere gli sprite, il valore 1 provoca la comparsa dello sprite in questione.
Questo è il quarto modulo del programma:

```
1290 GET KP$
1300 IF KP$ = "" THEN 1290
```

La linea 1290 legge dalla tastiera del vostro Commodore 64, e la linea 1300 provoca l'interruzione del programma quando viene letto un qualsiasi carattere da tastiera, altrimenti ritorna ad eseguire in ciclo l'istruzione 1290.
È sempre un'ottima abitudine lasciare tutto come lo si è trovato, specialmente programmando. Le linee 1350-1380 riportano i registri alterati del VIC-II al valore iniziale 0:

```
1350 POKE VIC+21,0 :REM  INVERTE L'ORDINE
1360 POKE VIC+39,0 :REM  USATO PER INIZIALIZZARE
1370 POKE VIC+1,0  :REM  I CONTROLLI
1380 POKE VIC,0    :REM  DELLO SPRITE
```

Notate come l'ordine con cui vengono azzerati i registri sia inverso a quello con cui vengono modificati.

ALCUNE PROVE

Uno dei modi migliori di imparare ad usare gli sprite è quello di cambiare alcuni dei valori assegnati alle variabili e vedere quali sono i risultati. Vi suggeriamo di provare le seguenti modifiche:
- Cambiate il numero del codice inviato dalla riga 1110.
- Cambiate la posizione orizzontale e verticale assegnata nelle righe 1210 e 1220.
- Cambiate il codice del colore della linea 1230.

ULTERIORI INFORMAZIONI SUL POSIZIONAMENTO DEGLI SPRITE

Quando posizionate uno sprite, dite al calcolatore dove posizionare l'angolo in alto a sinistra dello sprite. Il video del Commodore 64 può

H = 24
V = 29
H = 320
V = 29
H = 0
V = 50
H = 24
V = 50
H = 172
V = 50
H = 320
V = 50
H = 344
V = 50
Area dello schermo visibile
H = 24
V = 139
H = 172
V = 139
H = 320
V = 139
H = posizione orizzontale dell'angolo a sinistra in alto
V = posizione verticale dell'angolo a sinistra in alto
H = 0
V = 229
H = 24
V = 229
H = 172
V = 229
H = 320
V = 229
H = 344
V = 229
H = 24
V = 250
H = 320
V = 250

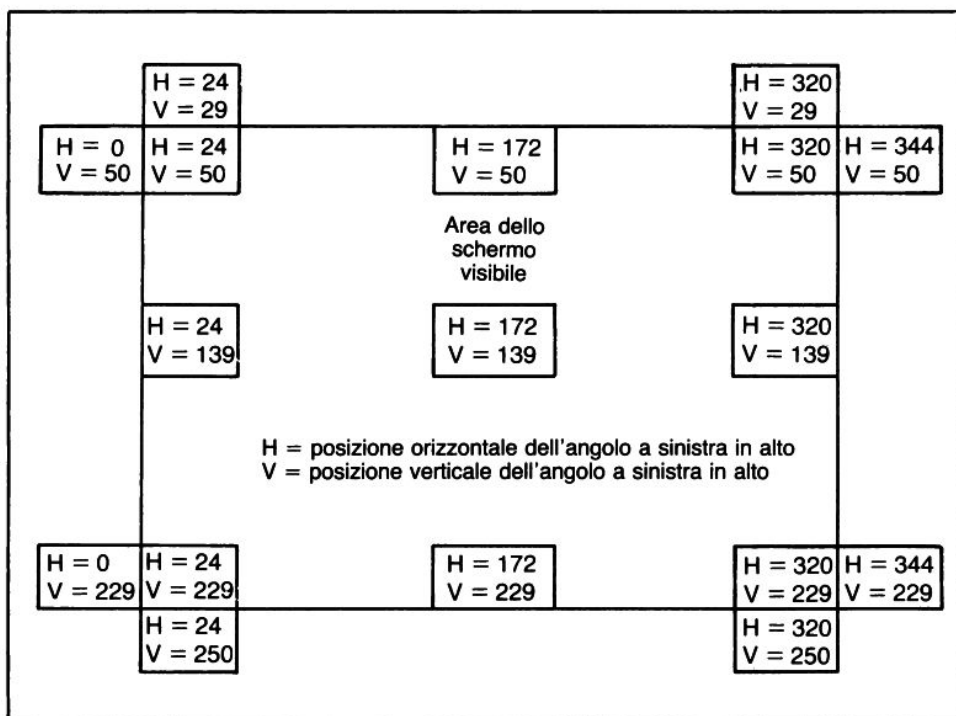



FIG. 1.10. Alcune posizioni in cui piazzare uno sprite.

posizionare uno sprite all'interno di uno spazio di 320 posizioni orizzontali e 200 verticali, mentre i registri del VIC-II possono indirizzare lo sprite in uno spazio di 512 per 256 posizioni. In questo modo potete muovere uno sprite sullo schermo con regolarità.
Guardate la figura 1.10: ci mostra le posizioni estreme in cui potete collocare il vostro sprite. Per esempio, le posizioni verticali minori di 29 collocano lo sprite al di fuori del campo visivo. Le posizioni orizzontali comprese tra 24 e 320 lo collocano completamente all'interno dell'area orizzontale visualizzata e così via.

UNO SPRITE YO-YO

Giochiamo un po'. Aggiungiamo al programma della figura 1.9 le linee riportate nella figura 1.11. Questo comporterà, oltre ad alcune variazioni nelle linee già esistenti, aggiunte di nuove linee. State attenti ad usare i numeri della figura per le linee che aggiungete. Quando avete finito salvate il programma e provatelo.
Come può lo sprite muoversi come uno yo-yo? Leggete attentamente le righe 1254-1256:

```
1254 FOR VP = 80 TO 200
1255 :    POKE VIC+1,VP
1256 NEXT VP
```

Questo ciclo dice al calcolatore di cambiare la posizione verticale dello sprite da 80 a 200 di una posizione alla volta. Lo sprite si muove verso il

```
1000 REM *** UNO SPRITE YO-YO ***
1220 POKE VIC+1,80   :REM POSIZ. VERTICALE
1251 :
1252 REM ** GIU', POI SU
1253 :
1254 FOR VP = 80 TO 200
1255 :    POKE VIC+1,VP
1256 NEXT VP
1257 :
1258 FOR VP = 199 TO 81 STEP -1
1259 :    POKE VIC+1,VP
1260 NEXT VP
1261 :
1262 :
1300 IF KP$ = "" THEN 1254
```

FIG. 1.11. Modifiche da apportare al vostro programma per ottenere "Uno sprite Yo-Yo".

basso. Quindi le linee da **1258** a **1260** lo spostano di nuovo verso l'alto di una posizione alla volta, da **199** a **81**:

```
1258 FOR VP = 199 TO 81 STEP -1
1259 :    POKE VIC+1,VP
1260 NEXT VP
```

Lo sprite si muove verso l'alto. Infine la riga **1300** dice al calcolatore di ritornare a eseguire le istruzioni dalla riga **1254**, se nel frattempo non è stato premuto nessun tasto.

```
1300 IF KP$ = "" THEN 1254
```

LAVORARE CON 512 POSIZIONI ORIZZONTALI

I lettori più attenti avranno notato, leggendo il paragrafo precedente e osservando la figura 1.10, che mentre i registri del Commodore 64 possono immagazzinare solo numeri compresi tra 0 e 255, il VIC-II risolve questo problema mettendo a disposizione 2 registri per ogni posizione orizzontale dello sprite. Il secondo registro è un registro in miniatura e può contenere solo uno zero oppure un uno. Se si vuole che uno sprite si trovi in una posizione maggiore di 255, si deve introdurre un uno nel secondo registro orizzontale dello sprite. La posizione dello sprite sarà 256 più il numero che c'è nel suo primo registro orizzontale. Per esempio: se il primo registro orizzontale di uno sprite contiene il numero 33 e il secondo registro orizzontale contiene il numero 1, lo sprite si troverà nella posizione 289 (256 + 33). Se il secondo registro orizzontale contiene uno 0, la posizione dello sprite è basata esclusivamente sul numero contenuto nel suo primo registro. La figura 1.12 mostra alcuni esempi che indicano come la posizione orizzontale di uno sprite possa andare da 0 a 511.

MOVIMENTI LATERALI DI UNO SPRITE

Considerate il primo programma, "Un semplice sprite", il cui listato è in figura 1.9. Caricate il primo programma, inserite i cambiamenti e le istruzioni ulteriori della figura 1.13, salvate il nuovo programma e fatelo girare. Prima di affrontare una discussione dettagliata sul funzionamento del nuovo programma, faremo un'escursione nel mondo della verità.

Se il primo registro orizzontale di uno sprite è posto a...	... e il suo secondo registro orizzontale è posto a...	... allora lo sprite si troverà nella posizione orizzontale
0	0	0
24	0	24
125	0	125
255	0	255
0	1	256
20	1	276
64	1	320
88	1	344
255	1	511

FIG. 1.12. Inizializzazione dei registri orizzontali per alcune posizioni di sprite tra 0 e 511.

Una codifica per vero e falso

Quando cercate di muovere uno sprite in una posizione orizzontale, dovrete prima chiedere se l'affermazione seguente è vera o falsa: "La nuova posizione è maggiore (>) di 255". Secondo la risposta ottenuta

```
1000 REM *** SPRITE DI LATO
1210 POKE VIC,170    :REM POSIZ. ORIZZONTALE
1220 POKE VIC+1,139 :REM POSIZ. VERTICALE
1251 :
1252 REM ** A DESTRA, POI A SINISTRA
1253 :
1254 FOR HP = 64 TO 280 STEP 2
1255 :    SF = (HP > 255)
1256 :    POKE VIC,HP + (SF * 256)
1257 :    POKE VIC+16, SF * (-1)
1258 NEXT HP
1259 :
1260 FOR HP = 278 TO 66 STEP -2
1261 :    SF = (HP > 255)
1262 :    POKE VIC, HP + (SF * 256)
1263 :    POKE VIC+16, SF * (-1)
1264 NEXT HP
1265 :
1266 :
1300 IF KP$ = "" THEN 1254
1400 END
```

FIG. 1.13. Cambiamenti e istruzioni ulteriori che trasformano il programma "Uno sprite semplice" nel programma "Sprite di lato".

metterete numeri diversi nei registri di posizione orizzontale dello sprite. Nel Commodore 64 Basic potete fare una domanda del tipo vero-falso e dare alla risposta una codifica particolare.
La codifica per "vero" è −1, mentre per "falso" è 0. Un esempio in Basic:

100 LET AN = (5 > 3).

Poiché 5 *è* maggiore di 3, l'espressione (5 > 3) è vera. Alla variabile AN sarà assegnato il valore −1. Esempio:

200 LET XZ = (36 = 21).

Poiché 36 è diverso da 21 l'espressione (36 = 21) è falsa e ad XZ è assegnato il valore 0.

Ritorniamo al programma: ci muoviamo a destra

Osserviamo come siamo riusciti a spostare lo sprite da una parte all'altra. Le linee 1210-1220 erano cambiate:

```
1210 POKE VIC,170   :REM POSIZ. ORIZZONTALE
1220 POKE VIC+1,139 :REM POSIZ. VERTICALE
```

Questo porta lo sprite in una nuova posizione. Ora osserviamo le linee 1254-1258:

```
1254 FOR HP = 64 TO 280 STEP 2
1255 :    SF = (HP > 255)
1256 :    POKE VIC,HP + (SF * 256)
1257 :    POKE VIC+16, SF * (-1)
1258 NEXT HP
```

Le linee 1254 e 1258 stabiliscono un ciclo che farà variare la posizione orizzontale dello sprite da 64 fino a 280, con incrementi di due. Ogni volta nel ciclo la linea 1255 calcolerà se la nuova posizione è maggiore di 255. Dalla risposta verrà poi stabilita la nuova posizione. Per esempio, se HP ha valore 125, l'istruzione 1255 fisserà SF (fattore dimensionale) a 0. La linea 1256 assegnerà al primo registro orizzontale dello sprite 125 + (0*256), che è ancora 125. La linea 1257 attribuirà al secondo registro orizzontale dello sprite – 1 × 0, cioè 0. Questi sono i corretti assegnamenti per una posizione minore di 256.

Proviamo la validità di queste formule per una posizione maggiore di 255. Supponiamo che HP abbia il valore 256, allora la linea 1257 attribuirà così al primo registro 276 + (− 1*256), cioè 276-256, ovvero 20. L'istruzione 1257 assegnerà al secondo registro orizzontale il valore − 1 * − 1, cioè 1. Ancora una volta le formule assegnano il valore corretto nei registri della posizione orizzontale.

Ci muoviamo a sinistra

Se non vi sono chiare le spiegazioni delle linee 1254-1258, rileggete gli ultimi due paragrafi e poi provate a ricavarvi le formule a mano con alcuni valori che troverete nella figura 1.12. Ora osservate le istruzioni 1260-1264:

```
1260 FOR HP = 278 TO 66 STEP -2
1261 :    SF = (HP > 255)
1262 :    POKE VIC, HP + (SF * 256)
1263 :    POKE VIC+16, SF * (-1)
1264 NEXT HP
```

Questa volta il nostro ciclo ci condurrà dalla posizione 278 alla posizione orizzontale 66, ancora con incrementi di due passi. Lo sprite si muoverà a sinistra. Le istruzioni 1261 e 1263 sono esattamente come le linee 1255-1257. Assegnare una nuova posizione ai registri è identico, indipendentemente dal fatto che vi muoviate verso destra o verso sinistra. Infine, abbiamo cambiato la linea 1300 per ritornare all'inizio della sezione del programma che riguarda i movimenti laterali.

```
1300 IF KP$ = "" THEN 1254
```

```
1000 REM *** DISEGNA UNO SPRITE ***
1105 :    READ SPDTA
1110 :    POKE N, SPDTA
1121 :
1122 DATA   0,  60,   0,   0,  36,   0
1123 DATA   0, 102,  24,   0, 102,  56
1124 DATA   0,  36,  56,   0,  60,  16
1125 DATA   0,  24,  16,   0,  24,  16
1126 DATA  15, 255, 240,   8, 126,   0
1127 DATA   8, 126,   0,   8,  24,   0
1128 DATA  28,  24,   0,  28,  24,   0
1129 DATA  24,  60,   0,   0,  60,   0
1130 DATA   0,  36,   0,   0,  36,   0
1131 DATA   0,  36,   0,   3, 231, 192
1132 DATA   3, 231, 192
1230 POKE VIC+39,1  :REM COLORE BIANCO
```

FIG. 1.14. Cambiamenti ed istruzioni ulteriori che trasformano il programma "Un semplice sprite" nel programma "Disegna uno sprite".

CANCELLAZIONE DI UN QUADRATO

Introduciamo in questo semplice disegno di sprite un carattere più interessante. Caricate il primo programma il cui listato è in figura 1.9, introducete le nuove istruzioni e i cambiamenti elencati in figura 1.14. Quando avete finito la solita procedura per salvare il programma e lo avete fatto girare, il vostro piccolo quadrato è sparito e viene introdotto il carattere che era stato disegnato e codificato nella figura 1.7. Osservate il nuovo ciclo che carica i dati dello sprite (le istruzioni 1100-1120):

```
1100 FOR N = 896 TO 958
1105 :   READ SPDTA
1110 :   POKE N, SPDTA
1120 NEXT N
```

Prima attibuivamo a ogni locazione di memoria il valore 255, che faceva accendere un pixel; ora stiamo utilizzando una istruzione di lettura (linea 1105) per ottenere la codifica del pixel da una serie di istruzioni DATA. Ogni codifica viene letta nella variabile SPDTA. Poi il valore di SPDTA viene introdotto nella memoria.
Ora guardate le undici istruzioni DATA: vengono elencati i 63 codici calcolati nella figura 1.7. Notate l'ordine in cui la codifica è introdotta: riga per riga dall'alto verso il basso e in ogni riga da sinistra a destra.
Infine, la nuova versione dell'istruzione 1230 cambia il colore allo sprite, attribuendogli il colore bianco.
Date le imperfezioni dei televisori a colori e del circuito di visualizzazione del Commodore 64 rispetto a certi sfondi, colori diversi assumono gradi diversi di definizione. Dovrete esercitarvi un po' per trovare le combinazioni che preferite. Io di solito inizio con sprite bianchi su sfondo nero e poi procedo con le mie elaborazioni.

RISOLUZIONE DI DUE PROBLEMI

Per quanto riguarda l'ultimo programma, nascono due problemi.
Primo, lo sprite è troppo piccolo per mostrare i propri dettagli. Secondo, è il mio disegno e non il vostro.
Risolviamo il secondo problema. Ritorniamo alla pagina : avete disegnato alcuni sprite e avete calcolato i 63 numeri di codifica per il vostro sprite preferito. Ora utilizzerete questa informazione. Caricate l'ultimo programma "Disegnare uno sprite", listate le istruzioni 1122-

```
1000 REM *** UNO SPRITE PIU' GRANDE ***
1233 POKE VIC23,1  :REM AMPLIAMENTO VERTICALE
1236 POKE VIC+29,1  :REM AMPLIAMENTO ORIZZONTALE
1353 POKE VIC+29,0
1356 POKE VIC+23,0
```

FIG. 1.15. Cambiamenti ed istruzioni ulteriori addizionali che trasformano un programma "Disegna uno sprite" in un programma "Uno sprite più grande".

1132, usate lo screen editor del Commodore per cambiare la codifica dei pixel con quella a cui si era risaliti a pagina .
Per quanto riguarda il primo problema, il VIC-II ci permette di espandere uno sprite orizzontalmente e verticalmente. I dettagli sono maggiormente visibili in uno sprite ingrandito. Introducete soltanto le 5 linee elencate in figura 1.15. Ricordate di salvare il vostro programma e fatelo girare.

ESPANSIONE DELLO SPRITE E REGISTRI DI ESPANSIONE

Uno sprite può diventare due volte più largo sullo schermo, due volte più alto o entrambe le cose. Ciò che dovete fare è dire a VIC-II che tipo di espansione desiderate. Il trentesimo registro di VIC-II, contenuto in VIC + 29, determina un'espansione orizzontale per tutti gli otto sprite. Introducendo 1 nel registro, lo sprite 0 raddoppia la sua larghezza. Se introducete 0 nel registro, lo sprite 0 mantiene la larghezza normale. Il ventiquattresimo registro di VIC-II, contenuto in VIC + 23, determina un'espansione verticale per tutti gli 8 sprite. Introducendo 1 nel registro, lo sprite 0 raddoppierà la propria altezza. Introducendo lo 0 lo sprite mantiene la sua altezza normale. Quando uno sprite ingrandito è posto sullo schermo, i numeri nei suoi registri di posizione orizzontale e verticale determinano ancora la locazione del suo angolo sinistro superiore. La figura 1.16 mostra come questo accada mettendo lo sprite in posizioni importanti dello schermo. Paragonate questa figura alla figura 1.10.
Nell'ultimo programma, "Uno sprite più grande", le istruzioni 1233 e 1236 hanno assegnato 1 a entrambi i registri di espansione, duplicando lo sprite 0:

```
1233 POKE VIC23,1   :REM AMPLIAMENTO VERTICALE
1236 POKE VIC+29,1   :REM AMPLIAMENTO ORIZZONTALE
```

Alla fine del programma, le istruzioni 1353 e 1356 hanno riportato lo sprite alla sua dimensione iniziale riassegnando gli 0:

```
1353 POKE VIC+29,0
1356 POKE VIC+23,0
```

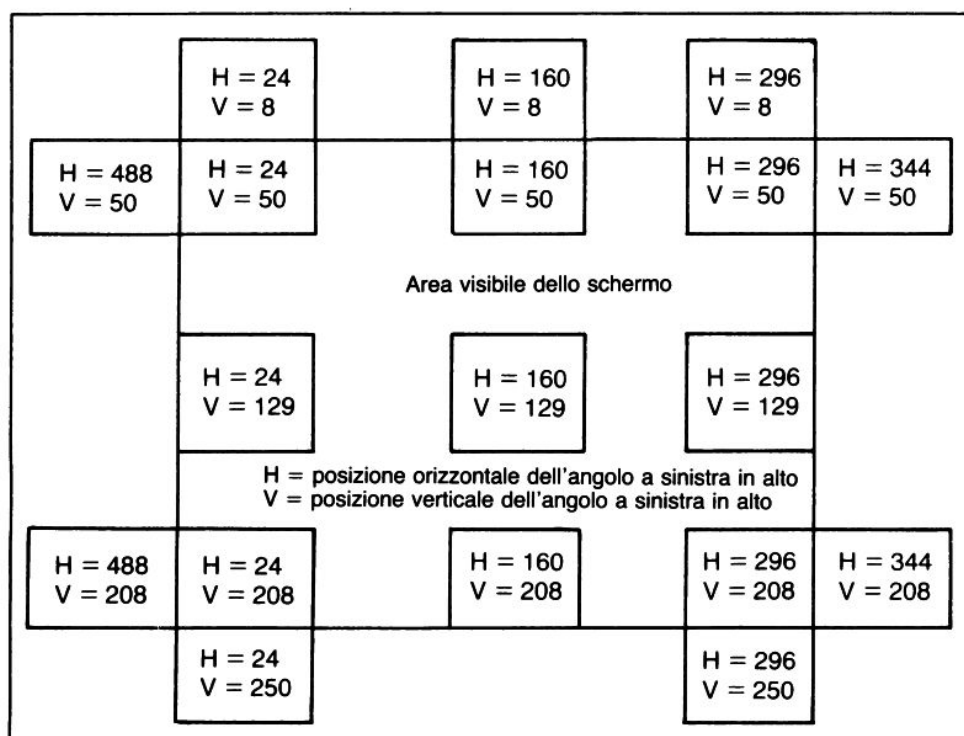



FIG. 1.16. Alcune importanti inizializzazioni di posizioni orizzontali e verticali per sprite di dimensioni doppie.

CHE COSA ABBIAMO IMPARATO

In questo capitolo abbiamo imparato:

- Che cosa sono un pixel e uno sprite
- Come disegnare uno sprite e come trasformare il disegno in 63 numeri codificati
- Come caricare la codifica di uno sprite e inizializzare i registri del VIC-II per rappresentare un semplice sprite sullo schermo
- Come inizializzare la posizione, il colore e la dimensione di uno sprite
- Come muovere uno sprite lateralmente o verticalmente.

ESERCIZI

Per impadronirvi meglio di queste nuove nozioni, affrontate il test e svolgete gli esercizi di programmazione proposti. Scrivete poi alcuni programmi utilizzando le idee espresse nel capitolo.

Test

Le risposte sono date più avanti.
1. Uno sprite è una configurazione di 504 ....
2. Quando si codifica una figura di sprite, si divide ognuna delle 21 righe in tre gruppi di .... pixel.
3. Per disegnare uno sprite bisogna caricare codici di 63 numeri, poi inizializzare .... nel ....
4. Quando si posiziona uno sprite, in pratica si comunica al VIC-II dove mettere l'angolo .... dello sprite.
5. Per muovere uno sprite dall'alto verso il basso, si deve cambiare il valore della posizione .... dello sprite.
6. Si usano .... registri per fissare una locazione orizzontale dello sprite, perché esistono .... posizioni possibili.
7. Nella seguente istruzione per il Commodore 64, a TV verrà assegnato il valore .... :

```
10 LET TV = (17 < 5)
```

8. Riscrivete l'istruzione 1230 del programma "Disegnare uno sprite" perché lo sprite assuma il colore giallo. L'appendice F vi aiuterà.
1230 ....
9. Uno sprite può essere ingrandito sia .... , sia .... , sia in entrambe le direzioni.

Esercizi di programmazione

Tutti questi programmi possono essere stesi sull'esempio del programma "Un semplice sprite" di fig. 1.9. Se preferite potete utilizzare varie istruzioni del programma. Le possibili soluzioni sono elencate più avanti. Ovviamente, tutto ciò che gira è corretto.

1) Scrivete un programma che faccia muovere lo sprite secondo un percorso rettangolare.

2) Fate che lo sprite cambi i colori ogni tanto.
3) Fate in modo che lo sprite passi ciclicamente per le 4 dimensioni possibili: normale, ingrandito orizzontalmente, ingrandito verticalmente e ingrandito in entrambe le direzioni.

Risposte al test

1. pixel
2. otto
3. registri; VIC-II
4. superiore sinistro
5. verticale
6. due; 512
7. zero
8. 1230 POKE VIC + 39,7 :REM Il colore è giallo
9. orizzontalmente; verticalmente

Soluzioni possibili agli esercizi di programmazione

Le mie tre soluzioni sono basate sul programma "Uno sprite semplice" il cui listato è rappresentato in figura 1.9. Qui sono elencate le istruzioni da aggiungere o cambiare a quel programma per risolvere l'esercizio.

1. Caricate il programma "Uno sprite semplice" e introducete queste istruzioni:

```
1000 REM *** MOVIMENTO RETTANGOLARE ***
1210 POKE VIC,82      :REM POSIZ. ORIZZONTALE
1220 POKE VIC+1,100 :REM POSIZ. VERTICALE
1241 :
1242 :
1243 REM ** SPOSTA A DESTRA, GIU', A SINISTRA,
1244 REM    POI DI NUOVO AL PUNTO DI PARTENZA
1245 :      (LE MOSSE ORIZZONTALI SALTANO
1246 REM    DI 3 PER CREARE CORRISPONDENZA
1247 REM     CON LE VELOCITA' VERTICALI)
1240 :
1249 FOR HP = 84 TO 261 STEP 3 :REM RT.
1250 :    SF = (HP > 255)
1251 :    POKE VIC, HP + (SF * 256)
1252 :    POKE VIC+16, SF * (-1)
1253 NEXT HP
1254 :
1255 FOR VP = 101 TO 179        :REM GIU'
1256 :    POKE VIC+1, VP
1257 NEXT VP
1258 :
1259 FOR HP = 258 TO 87 STEP -3 :REM LF
```

```
1260 :    SF = (HP > 255)
1261 :    POKE VIC, HP + (SF * 256)
1262 :    POKE VIC+16, SF * (-1)
1263 NEXT HP
1264 :
1265 FOR VP = 178 TO 100 STEP -1 :REM ↑
1266 :    POKE VIC+1, VP
1267 NEXT VP
1268 :
1269 :
1300 IF KP$ = "" THEN 1249

READY.
```

2. Caricate il programma "Uno sprite semplice" e introducete queste istruzioni:

```
1000 REM *** CAMBIA COLORI  ***
1071 REM ** FISSA IL COLORE DI PARTENZA
1072 :
1073 OC = 13        :REM INIZIA CON IL VERDE
1074 :
1075 :
1230 POKE VIC+39,OC :REM COLORE DI PARTENZA
1251 :
1252 REM ** CHANGE COLORS
1253 :
1254 FOR DELAY = 1 TO 500 : NEXT
1255 NC = OC + 1           :REM NUOVO COLORE
1256 IF NC=16 THEN NC=0    :REM I COLORI VANNO FINO A 15
1257 POKE VIC+39,NC        :REM LI INSERISCE
1258 OC = NC               :REM VECCHIO COLORE
1259 :
1300 IF KP$ = "" THEN 1254
```

3. Caricate il programma "Uno sprite semplice" e introducete queste istruzioni:

```
1000 REM *** CICLO DI CRESCITA ***
1251 :
1252 REM ** ESPANSIONE ORIZZONTALE **
1253 :
1254 FOR DELAY = 1 TO 400   NEXT
1255 POKE VIC+29,1
1256 :
1257 :
1258 REM ** CONTRAZIONE ORIZZONTALE **
1259 REM     ED ESPANSIONE VERTICALE
1260 :
1261 FOR DELAY = 1 TO 400   NEXT
1262 POKE VIC+29,0
1263 POKE VIC+23,1
1264 :
1265 :
1266 REM ** ESPANSIONE ORIZZONTALE
1267 :
1268 FOR DELAY = 1 TO 400   NEXT
1269 POKE VIC+29,1
1270 :
```

```
1271 :
1272 REM ** CONTRAZIONE ORIZZONTALE **
1273 REM    E CONTRAZIONE VERTICALE
1274 :
1275 FOR DELAY = 1 TO 400   NEXT
1276 POKE VIC+29,0
1277 POKE VIC+23,0
1278 :
1279 :
1280 REM ** ASPETTA LA PRESSIONE DI UN TASTO PER TERMINARE
1281 :
1300 IF KP$ = "" THEN 1254
```

CAPITOLO DUE

Più di uno sprite

Questo capitolo illustra come trattare più sprite sullo schermo. Imparerete a usare lo stesso blocco di dati di sprite per costruirne diversi, e ad alterare il modo in cui sono rappresentati i dati. Inoltre imparerete a inserire differenti sprite sullo schermo e, infine, a muoverli con continuità.

CLONI SEMPLICI

Il Commodore 64 permette la formazione di diversi sprite che usano lo stesso blocco di pixel. Se inizializzate gli sprite in diversi punti dello schermo e ne conservate la grandezza essi risultano delle semplici copie, dei "cloni".
La figura 2.1 dà il listato del programma "Cloni semplici".
Questo programma disegnerà quattro copie di uno stesso sprite, quello raffigurato nella figura 2.2.
Il programma è molto simile a quello del capitolo 1 per disegnare un unico sprite. La differenza fondamentale è che in questo caso bisogna inizializzare i puntatori, le locazioni e i colori per quattro sprite. Caricate il programma.
Salvatelo su nastro o su disco e fatelo girare. Quando la vostra esecuzione sarà finita tornate a leggere le spiegazioni.
Nel capitolo 1, abbiamo visto come la locazione di memoria 2040 venga normalmente usata per comunicare al VIC-II dove si trovano i codici dei pixel dello sprite 0. Le locazioni 2041-2047 sono usate normalmente

```
1000 REM *** CLONI SEMPLICI ***
1010 :
1020 :
1030 REM ** PROVVEDE UN FEEDBACK SULLO SCHERMO
1040
1050 PRINT "[CLR][DOWN][DOWN][DOWN][DOWN][DOWN][DOWN][DOWN][DOWN]STO PENSANDO";
1060 :
1070 :
1080 REM ** CARICA I DATI DELLO SPRITE
1090 :
1100 FOR N = 896 TO 958
1110 :   READ SPDTA
1120 :   POKE N, SPDTA
1130 NEXT N
1140 :
1150 DATA   6, 102,  96,   6, 102,  96
1160 DATA   6, 102,  96,   7, 255, 224
1170 DATA  15, 255, 240,  28,   0,  56
1180 DATA  56,   0,  28, 113, 195, 142
1190 DATA 225, 195, 135, 193, 195, 131
1200 DATA 192,   0,   3, 192,   0,   3
1210 DATA 192,  60,   3, 192,   0,   3
1220 DATA 204,   0,  51, 198,   0,  99
1230 DATA 195, 255, 195, 192,   0,   3
1240 DATA 224,   0,   7, 127, 255, 254
1250 DATA  63, 255, 252
1260 :
1270
1280 REM ** REGOLA I CONTROLLI DEGLI SPRITE
1290 :
1300 PRINT "[CLR]"       :REM PULISCE LO SCHERMO
1310 VIC = 53248      :REM CHIP GRAFICO
1320 :
1330 POKE 2040,14     :REM PUNTATORE DATI SPRITE 0
1340 POKE 2041,14     :REM PUNTATORE DATI SPRITE 1
1350 POKE 2042,14     :REM PUNTATORE DATI SPRITE 2
1360 POKE 2043,14     :REM PUNTATORE DATI SPRITE 3
1370 :
1380 POKE VIC,98      :REM POSIZ. ORIZZONTALE SPRITE 0
1390 POKE VIC+2,246   :REM POSIZ. ORIZZONTALE SPRITE 1
1400 POKE VIC+4,98    :REM POSIZ. ORIZZONTALE SPRITE 2
1410 POKE VIC+6,246   :REM POSIZ. ORIZZONTALE SPRITE 3
1420 :
1430 POKE VIC+1,95    :REM POSIZ. VERTICALE SPRITE 0
1440 POKE VIC+3,95    :REM POSIZ. VERTICALE SPRITE 1
1450 POKE VIC+5,184   :REM POSIZ. VERTICALE SPRITE 2
1460 POKE VIC+7,184   :REM POSIZ. VERTICALE SPRITE 3
1470 :
1480 POKE VIC+39,1    :REM SPRITE 0 BIANCO
1490 POKE VIC+40,3    :REM SPRITE 1 CYAN
1500 POKE VIC+41,5    :REM SPRITE 2 VERDE
1510 POKE VIC+42,7    :REM SPRITE 3 GIALLO
1520 :
1530 POKE VIC+21,15   :REM SPRITE 0-3 ATTIVI
1540 :
1550 :
1560 REM ** ASPETTA LA PRESSIONE DI UN TASTO PER FINIRE
1570 :
1580 GET KP$
1590 IF KP$ = "" THEN 1580
1600 :
1610 :
1620 REM ** REINIZIALIZZA I CONTROLLI DEGLI SPRITE
```

```
1630 :
1640 POKE VIC+21,0
1650 :
1660 END
```

FIG. 2.1. Listato del programma "Cloni semplici".

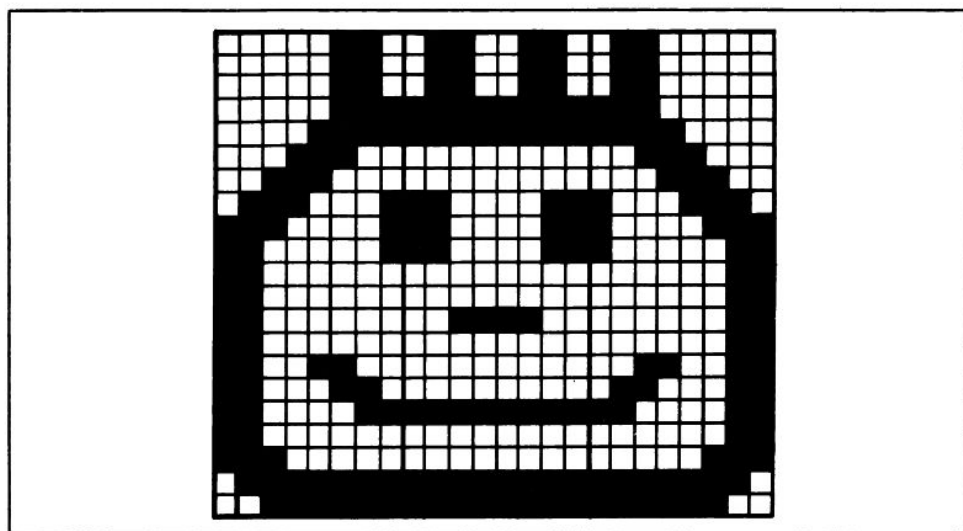



FIG. 2.2. Un disegno di un semplice sprite, pronto per la clonazione.

per comunicare dove si trovano i codici del pixel per gli sprite da 1 a 7. La figura 2.3 mostra quale locazione di memoria punta ai dati per un particolare sprite.

Locazione di memoria →	2040	2041	2042	2043	2044	2045	2046	2047
Punta ai dati dei pixel per lo sprite numero →	0	1	2	3	4	5	6	7

FIG. 2.3. Locazioni di memoria per i puntatori ai dati dello sprite.

Inizializzazione dei 4 sprite

Affrontiamo ora la parte più importante del programma "Cloni semplici". Le istruzioni 1000-1310 vi saranno ormai familiari. Si ha una retroazione sul video e il pixel è caricato nelle locazioni di memoria 896-958; il video viene pulito, e la variabile VIC viene inizializzata con l'indirizzo iniziale del VIC-II. Le istruzioni 1330-1360 sono i primi segni di qualcosa di nuovo:

```
1330 POKE 2040,14     :REM PUNTATORE DATI SPRITE 0
1340 POKE 2041,14     :REM PUNTATORE DATI SPRITE 1
1350 POKE 2042,14     :REM PUNTATORE DATI SPRITE 2
1360 POKE 2043,14     :REM PUNTATORE DATI SPRITE 3
```

Questo programma farà visualizzare quattro sprite. Ogni sprite prenderà i suoi dati dalle 63 locazioni di memoria che partono dalla locazione (14 × 64) ovvero 896. Poi le istruzioni 1380-1460 attribuiscono a ogni sprite una posizione orizzontale e verticale sul video:

```
1380 POKE VIC,98      :REM POSIZ. ORIZZONTALE SPRITE 0
1390 POKE VIC+2,246   :REM POSIZ. ORIZZONTALE SPRITE 1
1400 POKE VIC+4,98    :REM POSIZ. ORIZZONTALE SPRITE 2
1410 POKE VIC+6,246   :REM POSIZ. ORIZZONTALE SPRITE 3
1420 :
1430 POKE VIC+1,95    :REM POSIZ. VERTICALE SPRITE 0
1440 POKE VIC+3,95    :REM POSIZ. VERTICALE SPRITE 1
1450 POKE VIC+5,184   :REM POSIZ. VERTICALE SPRITE 2
1460 POKE VIC+7,184   :REM POSIZ. VERTICALE SPRITE 3
```

La locazione VIC + 3 (53248) è il primo registro di posizione orizzontale per lo sprite 0 e VIC + 1 (53249) è il registro di posizione verticale dello sprite 0. I successivi quattordici registri del VIC-II seguono lo stesso andamento, per gli altri sette sprite. VIC + 2 (53250) è il primo registro di posizione orizzontale per lo sprite 1, VIC + 3 (53251) è il suo registro di posizione verticale. Si continua così fino alla locazione VIC + 15 (53263) che è il registro di posizione verticale per lo sprite 7. L'appendice A fornisce tutti i dettagli.
I lettori curiosi potrebbero domandarsi che cosa accadrebbe se inserissimo un secondo registro orizzontale per ogni sprite. Ricordate che il secondo registro orizzontale di ogni sprite è un registro in miniatura, che può contenere solo un 1 o uno 0. Otto di questi registri in miniatura sono collocati in una sola locazione di memoria, che è VIC + 16 (53264). Più avanti in questo capitolo parleremo ancora di questi registri in miniatura.

Colorazioni e accensione degli sprites

Le istruzioni 1480-1510 attribuiscono un colore a ogni sprite:

```
1480 POKE VIC+39,1   :REM SPRITE 0 BIANCO
1490 POKE VIC+40,3   :REM SPRITE 1 CYAN
1500 POKE VIC+41,5   :REM SPRITE 2 VERDE
1510 POKE VIC+42,7   :REM SPRITE 3 GIALLO
```

Come avrete indovinato i registri che controllano il colore di ogni sprite si trovano in otto locazioni consecutive del VIC-II: da VIC + 39 (53287) a VIC + 46 (53294). Consultate ancora l'appendice A per ulteriori dettagli riguardo i registri del VIC-II e l'appendice F per quanto riguarda la codifica dei valori. Infine la linea 1530 visualizza i quattro sprite: 0, 1, 2 e 3.

```
1530 POKE VIC+21,15 :REM SPRITE 0-3 ATTIVI
```

Ma che cosa ha a che fare 15 con 4 o 0, 1, 2, 3? Per poterlo spiegare è opportuna una breve digressione su bit e byte.

Bit e byte

Vi ricordate quando abbiamo imparato a tradurre i disegni di pixel in una codifica? Abbiamo preso l'informazione a gruppi di 8 punti. Perché 8 e non 9 o 10, o 24? Il chip di un Commodore 64 può manipolare solo un numero alla volta e quel numero non può essere troppo grande. Infatti deve essere compreso in un intervallo tra 0 e 255. Inoltre, il numero deve essere rappresentato usando solo le cifre 0 e 1.
Un gruppo di otto cifre che siano solo 1 o 0 può rappresentare qualsiasi numero compreso fra 0 e 255. Questo sistema di numerazione è detto binario e ogni cifra, lo zero oppure l'uno, è un "bit".
Un gruppo di otto bit è detto "byte". Ogni locazione di memoria del Commodore, inclusi i registri di VIC-II, può immagazzinare un byte. Le figure 2.4 e 2.5 vi mostrano alcuni bit e byte.
Molte locazioni di memoria del VIC-II possono controllare le funzioni di otto sprite, assegnando a ogni sprite uno degli otto bit di quella locazione. Così, un bit può essere pensato come un registro in miniatura che controlla uno sprite.
Il registro nella locazione VIC + 21 (53269) è un controllo importante di accensione per gli otto sprite. Ogni sprite può essere acceso o spento, attivato o disattivato, giocando con questa locazione. Ogni bit è un registro in miniatura che accende o spegne uno sprite.

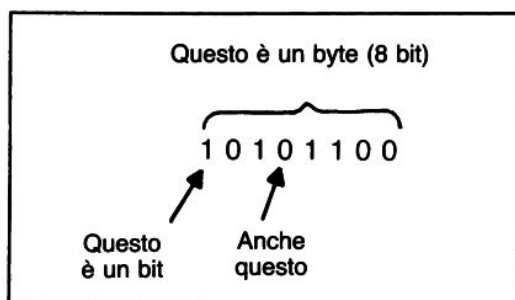



FIG. 2.4. Un byte è costituito da otto bit.

Questo numero decimale →	255	240	128	127	60	15	1	0
Si trasforma in questo byte binario →	11111111	11110000	10000000	01111111	00111100	00001111	00000001	00000000

FIG. 2.5. I byte binari, composti da 8 bit, possono rappresentare valori normali (in base 10) tra 0 e 255.

Gli otto bit in un byte sono numerati da 0 a 7. Nella locazione VIC + 21, il bit 0 controlla lo sprite 0, il bit 1 controlla lo sprite 1, e così via. Per attivare uno sprite particolare, basta inserire un 1 nel suo corrispondente bit in VIC + 21. Per disattivare uno sprite inserite uno 0 nel suo bit in VIC + 21.

Per disegnare i 4 sprite numerati da 0 a 3, dovete assegnare a VIC + 21 (il registro acceso/spento) un numero che avrà la cifra 1 nei bit 0, 1, 2, 3 e la cifra 0 nelle altre posizioni del byte. Sembra difficile, ma in realtà potete usare lo stesso diagramma usato per codificare un gruppo di otto pixel.

La figura 2.6 mostra un byte con i suoi 8 bit numerati da 0 a 7. A ogni bit viene anche assegnato un valore. Mettete degli 1 nelle posizioni dei bit degli sprite che volete accesi e 0 dove volete gli sprite spenti. Poi, sommando i valori dei bit che contengono 1, ricavate il numero da inserire in memoria per ottenere la giusta successione di 1 e 0.

La figura 2.7 mostra alcuni esempi. Guardiamo quello che compare nel programma "Cloni semplici". Si vogliono accendere gli sprite 0-3, così bisogna immagazzinare 1 nei bit 0-3. Si sommano i valori di quei bit (8 + 4 + 2 + 1) e si ottiene 15. Si è così risolto il mistero del 15.

Il resto del programma vi dovrebbe essere familiare. Le istruzioni 1580-1590 aspettano che voi digitiate. Quando digitate, l'istruzione

FIG. 2.6. Un byte con i suoi 8 bit numerati da 0 a 7. A ogni bit è associato il suo valore di posizione.

1640 inizializza i controlli dello sprite; comunque non tutti gli sprite hanno bisogno di essere inizializzati. Deve essere inizializzato a 0 il registro di acceso/spento nella locazione VIC + 21. Se avete ingrandito uno sprite orizzontalmente o verticalmente, al registro di ingrandimento dello sprite in VIC + 23 e VIC + 29 dovrebbe essere riassegnato 0. Così, non avrete la sorpresa di ottenere sprite allungati in modi inaspettati.

CLONI COMPLESSI

Sebbene usino gli stessi dati dei pixel, i 4 sprite nell'ultimo programma non sono esattamente uguali; ognuno appare sul video colorato diversamente. Potete farli sembrare ancora meno simili ingrandendoli in diversi modi.

Come abbiamo imparato a pagina 32, la locazione VIC + 29 fa in modo che vi sia espansione orizzontale per gli sprite. Ogni bit, nei byte immagazzinati in VIC + 29, controlla l'espansione orizzontale di uno sprite. Se volete l'ingrandimento orizzontale degli sprite 2 e 3, i bit 2 e 3 devono essere inizializzati a 1. Usando per i bit i valori mostrati in figura 2.6, potete trovare il numero da assegnare nel registro: 12 (8 + 4). Guardate la figura 2.8.

Il registro di VIC + 23 si occupa dell'espansione verticale per gli otto sprite, nello stesso modo. Se volete che gli sprite 1 e 3 si espandano verticalmente, per esempio, dovete dare il valore 1 ai bit 1 e 3 di quel registro. Sommando i valori dei bit, trovate il numero da assegnare a VIC + 23 cioè 10 (8 + 2). Guardate la figura 2.9.

Usiamo queste nuove istruzioni per cambiare il programma "Cloni semplici", carichiamolo nel computer e inseriamo le linee elencate in

Sprite attivi	Sprite non attivi	Byte del registro (ciascun bit controlla lo sprite con lo stesso numero)	Numero da inserire
	0–7	Valore del bit: 128 64 32 16 8 4 2 1 0 0 0 0 0 0 0 0 Numero del bit: 7 6 5 4 3 2 1 0	0
0	1–7	Valore: 128 64 32 16 8 4 2 1 0 0 0 0 0 0 0 1 Numero: 7 6 5 4 3 2 1 0	1= 1
0,1	2–7	Valore: 128 64 32 16 8 4 2 1 0 0 0 0 0 0 1 1 Numero: 7 6 5 4 3 2 1 0	1+2= 3
0–3	4–7	Valore: 128 64 32 16 8 4 2 1 0 0 0 0 1 1 1 1 Numero: 7 6 5 4 3 2 1 0	1+2+4+8= 15
0,2, 4,6	1,3, 5,7	Valore: 128 64 32 16 8 4 2 1 0 1 0 1 0 1 0 1 Numero: 7 6 5 4 3 2 1 0	1+4+16+ 64= 85
0-7	–	Valore: 128 64 32 16 8 4 2 1 1 1 1 1 1 1 1 1 Numero: 7 6 5 4 3 2 1 0	1+2+4+8+ 16+32+64 +128= 255

FIG. 2.7. In questi esempi, un registro che ha la dimensione di un byte usa i suoi 8 bit per accendere o spegnere gli sprite. In ogni caso i bit individuali sono inizializzati assegnando i valori nella colonna a destra.

figura 2.10 per trasformarlo nel programma "Cloni complessi". Salvate il nuovo programma e fatelo girare.

Ora abbiamo sul video quattro sprite, tutti basati sullo stesso insieme di dati relativi ai pixel, ma ognuno sembra molto diverso dagli altri. Ci siamo riusciti senza troppe difficoltà. Le nuove versioni delle istruzioni

Valore del bit →	128	64	32	16	8	4	2	1
Bit →	0	0	0	0	1	1	0	0
Numero del bit →	7	6	5	4	3	2	1	0

8 + 4 = 12

FIG. 2.8. Assegnando il valore 12 nel registro di espansione orizzontale si inizializzano a 1 i bit 2 e 3, causando l'espansione orizzontale degli sprite 2 e 3.

Numero del bit →	128	64	32	16	8	4	2	1
Bit →	0	0	0	0	1	0	1	0
Valore del bit →	7	6	5	4	3	2	1	0

8 + 2 = 10

FIG. 2.9. Assegnare 10 nel registro di espansione verticale significa dare valore 1 ai bit 1 e 3, causando l'espansione verticale degli sprite 1 e 3.

```
1000 REM *** CLONI COMPLESSI ***
1400 POKE VIC+4,86   :REM POSIZ. ORIZZONTALE SPRITE 2
1410 POKE VIC+6,234  :REM POSIZ. ORIZZONTALE SPRITE 3
1440 POKE VIC+3,85   :REM POSIZ. VERTICALE SPRITE 1
1460 POKE VIC+7,174  :REM POSIZ. VERTICALE SPRITE 3
1522 POKE VIC+23,10  :REM SPRITE 1 E 3 GRANDI
1524 POKE VIC+29,12  :REM SPRITE 2 E 3 LARGHI
1526 :
1642 POKE VIC+23,0
1644 POKE VIC+29,0
```

FIG. 2.10. Cambiamenti e istruzioni che trasformano "Cloni semplici" in "Cloni complessi".

1400, 1410, 1440 e 1460 muovono gli sprite. Poi le istruzioni 1522 e 1524 determinano le espansioni degli sprite usate prima come esempi:

```
1522 POKE VIC+23,10 :REM SPRITE 1 E 3 GRANDI
1524 POKE VIC+29,12 :REM SPRITE 2 E 3 LARGHI
```

Lo sprite 0 rimane nella sua dimensione normale. Lo sprite 1 diventa più alto. Lo sprite 2 diventa più largo. Lo sprite 3 viene espanso in ambedue le direzioni. Quando viene digitato un tasto segnala la fine del programma. Le istruzioni 1642 e 1644 riportano a 0 i registri di espansione.

IMMAGAZZINARE PIÙ DI UN BLOCCO DI DATI DI SPRITE

In molti casi si desidera avere sprite molto diversi l'uno dall'altro. Per questo si deve inserire un blocco di dati relativi ai pixel per ogni diversa immagine di sprite. Dove si devono mettere i 63 numeri per ogni immagine?
Se state usando non più di tre diverse immagini di sprite, potete inserire i dati in queste tre aree: locazioni di memoria 832-894, 896-958, 960-1022. Queste aree di memoria sono usate con il registratore del Commodore. Quindi sono perfettamente sicure, quando si rimane all'interno di un programma. I puntatori ai dati dello sprite nelle istruzioni 2040-2047 devono contenere l'indirizzo iniziale del blocco dei dati relativi ai pixel diviso per 64: per queste aree dunque i puntatori conterranno rispettivamente 13, 14 o 15.
Se avete bisogno di più di 3 blocchi di dati relativi ai pixel, usate le locazioni di memoria a partire dalla 12288. La figura 2.11 dà le locazioni con il numero del puntatore usato per ogni area. (Posizioni più "esotiche" sono accessibili ai programmatori avanzati che vogliano "giocare" con la mappa di memoria del Commodore 64, ma qui non ce ne occuperemo.)

Area di memoria da 63 byte →	12288 — 12350	12352 — 12414	12416 — 12478	12480 — 12542	12544 — 12606	12608 — 12670	12672 — 12734	12736 — 12798
Fissa il puntatore → a	192	193	194	195	196	197	198	199

FIG. 2.11. Aree per inserire i dati dello sprite con i valori appropriati del puntatore.

VISUALIZZARE DUE SPRITE DIVERSI

Immaginate un programma che visualizzi due immagini di sprite differenti sul video. Come differirà dal programma "Disegnare uno sprite" considerato nel capitolo 1?
In primo luogo dovrà caricare due blocchi di dati relativi ai pixel. Poi

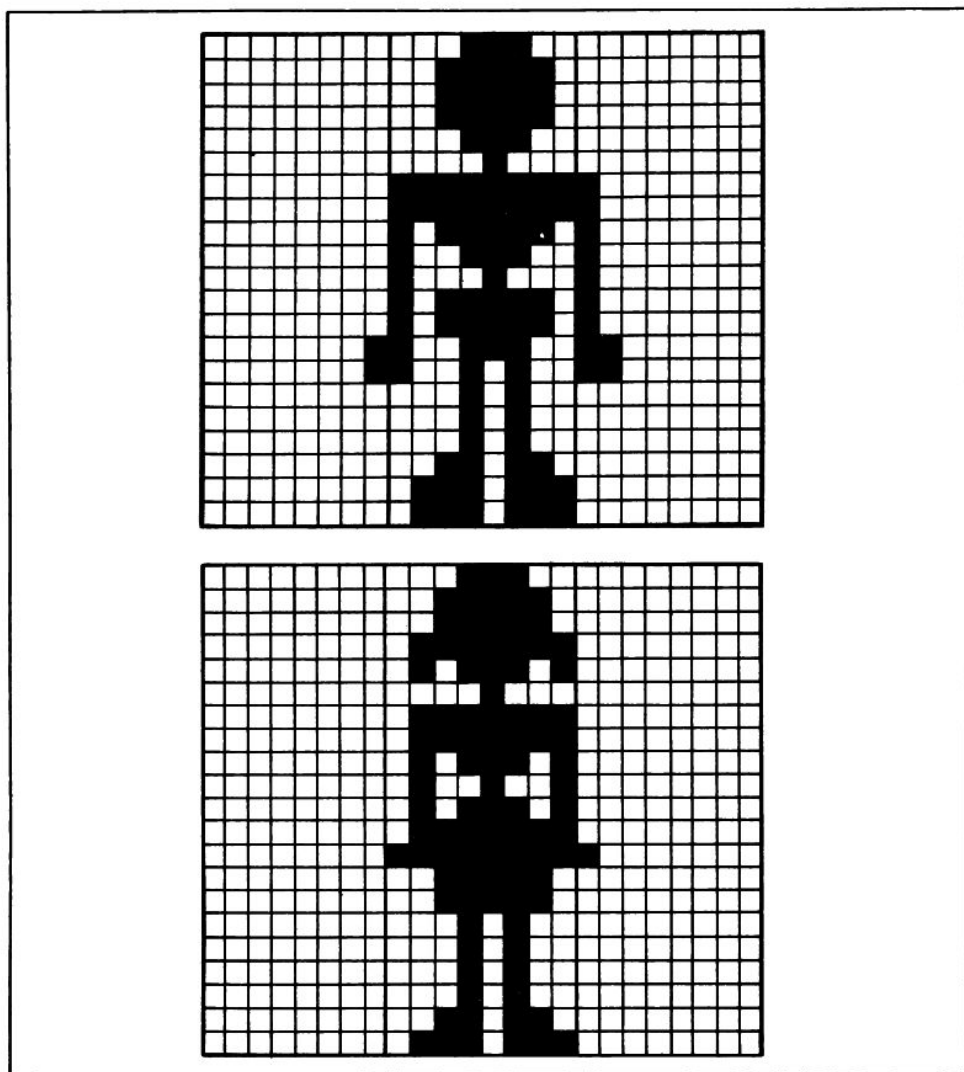



FIG. 2.12. I disegni di due nuovi sprite.

dovrà inizializzare i puntatori a **2040** e **2041** per puntare a due aree contenenti i dati relativi ai pixel. Dovrà inizializzare i registri del VIC-II per posizionare, colorare e dimensionare ogni sprite. Infine dovrà visualizzare entrambi gli sprite.

La figura 2.12 mostra i disegni di due nuovi sprite, mentre la figura 2.13 dà il listato del programma "Coppia di sprite", che li visualizza sullo schermo. Battete il programma, memorizzatelo e provate a eseguirlo. Modificatelo un po' per vedere che cosa succede, cambiando parti delle immagini e valori dei registri.

```
1000 REM *** COPPIA DI SPRITE ***
1010 -
1020 :
1030 REM ** PROVVEDE UN FEEDBACK SULLO SCHERMO
1040 :
1050 PRINT "[illegible]STO PENSANDO";
1060 :
1070 :
1080 REM ** CARICA I DATI DEGLI SPRITE
1090 :
1100 FOR N = 896 TO 958    :REM PRIMO
1110 :    READ SPDTA
1120 :    POKE N, SPDTA
1130 NEXT N
1140 PRINT ". ";            :REM FEEDBACK
1150 :
1160 FOR N = 960 TO 1022   :REM SECONDO
1170 :    READ SPDTA
1180 :    POKE N, SPDTA
1190 NEXT N
1200 PRINT ". ";            :REM FEEDBACK
1210 :
1220 DATA   0,  28,   0,   0,  62,   0
1230 DATA   0,  62,   0,   0,  62,   0
1240 DATA   0,  28,   0,   0,   8,   0
1250 DATA   0, 255, 128,   0, 255, 128
1260 DATA   0, 190, 128,   0, 156, 128
1270 DATA   0, 136, 128,   0, 190, 128
1280 DATA   0, 190, 128,   1, 156, 192
1290 DATA   1, 148, 192,   0,  20,   0
1300 DATA   0,  20,   0,   0,  20,   0
1310 DATA   0,  54,   0,   0, 119,   0
1320 DATA   0, 119,   0
1330 :
1340 DATA   0,  28,   0,   0,  62,   0
1350 DATA   0,  62,   0,   0, 127,   0
1360 DATA   0,  93,   0,   0,   8,   0
1370 DATA   0, 127,   0,   0, 127,   0
1380 DATA   0,  93,   0,   0,  73,   0
1390 DATA   0,  93,   0,   0, 127,   0
1400 DATA   0, 255, 128,   0,  62,   0
1410 DATA   0,  62,   0,   0,  20,   0
1420 DATA   0,  20,   0,   0,  20,   0
1430 DATA   0,  20,   0,   0,  54,   0
1440 DATA   0, 119,   0
1450 :
1460 :
```

```
1470 REM ** FISSA I CONTROLLI DEGLI SPRITE
1480 :
1490 PRINT "□";        :REM PULISCE LO SCHERMO
1500 VIC = 53248       :REM CHIP GRAFICO
1510 :
1520 POKE 2040,14      :REM PUNTATORE DATI SPRITE 0
1530 POKE 2041,15      :REM PUNTATORE DATI SPRITE 1
1540 :
1550 POKE VIC,124      :REM POSIZ. ORIZZONTALE SPRITE 0
1560 POKE VIC+2,173    :REM POSIZ. ORIZZONTALE SPRITE 1
1570 POKE VIC+1,150    :REM POSIZ. VERTICALE SPRITE 0
1580 POKE VIC+3,150    :REM POSIZ. VERTICALE SPRITE 1
1590 :
1600 POKE VIC+39,3     :REM SPRITE 0 CYAN
1610 POKE VIC+40,7     :REM SPRITE 1 GIALLO
1620 :
1630 POKE VIC+23,3     :REM AMBEDUE GLI SPRITE
1640 POKE VIC+29,3     :REM IN DIMENSIONE DOPPIA
1650 :
1660 POKE VIC+21,3     :REM ATTIVA AMBEDUE GLI SPRITE
1670 :
1680 :
1690 REM ** ASPETTA LA PRESSIONE DI UN TASTO PER FINIRE
1700 :
1710 GET KP$
1720 IF KP$ = "" THEN 1710
1730 :
1740 :
1750 REM ** REINIZIALIZZA I CONTROLLI DEGLI SPRITE
1760 :
1770 POKE VIC+21,0     :REM DISATTIVA GLI SPRITE
1780 POKE VIC+29,0     :REM E RIPORTA LE DIMENSIONI
1790 POKE VIC+23,0     :REM A QUELLE NORMALI
1800 :
1810 END
```

FIG. 2.13. Mostra il listato del programma "Coppia di sprite" che serve a visualizzare gli sprite sullo schermo.

TUTTO SULLA VOSTRA COPPIA DI SPRITE

Nulla vi dovrebbe sorprendere nel programma "Coppia di sprite". Vediamone comunque qualche punto! L'istruzione **1050** pulisce lo schermo e lo predispone per il nuovo inserimento; poi due cicli caricano i due blocchi di dati dei pixel. Il primo insieme di 63 numeri è posto nelle locazioni **896-958**. L'istruzione **1140** segnala che il primo blocco è inizializzato mettendo un punto dopo la parola "pensare". Il secondo insieme di 63 numeri è messo nelle locazioni **960-1022**. Poi l'istruzione **1200** segnala la fine di quel processo con un altro punto. I dati dei pixel sono stati calcolati utilizzando una copia della scheda di codifica dalla figura 1.6.

Il programma inizializza anche i puntatori dei dati e i registri del VIC-II. Ho deciso di raddoppiare le dimensioni degli sprite dato che erano così dettagliati. È difficile notare i dettagli a dimensione normale. L'istru-

zione 1600 accende gli sprite 0 e 1 inserendo 1 nei bit 0 e 1 di VIC + 21. Ritornate a pagina 42-43, se non vi è del tutto chiaro il motivo di questa scelta.
Le istruzioni 1700 e 1710 attendono che premiate un tasto per finire. Le istruzioni 1770-1790 inizializzano i registri di acceso/spento e di espansione.

MUOVERE PIÙ DI UNO SPRITE ALLA VOLTA

Esistono diverse tecniche che si possono usare per muovere più di uno sprite. Alcune sono facilmente programmabili, altre meno.
Alcune utilizzano molta memoria, altre poca. Alcune forniscono cammini semplici, altre complessi. Alcune permettono movimenti veloci e continui, altre movimenti lenti e scoordinati. Alcune infine sono molto chiare, altre difficili da comprendere. In questo paragrafo farò un solo esempio: non è troppo difficile ma dà buoni risultati.
Prenderemo due sprite dal programma "Coppia di sprite" e faremo sì che si rincorrano per il video. Programmeremo questo movimento cambiando e aggiungendo istruzioni al programma "Coppia di sprite", quindi caricatelo in macchina e inserite le linee indicate in figura 2.14. Memorizzate e fate girare il programma risultante.

Pensiamo al percorso

In questo programma, i due sprite gareggiano lungo un percorso quadrato centrato sullo schermo; gli angoli saranno lontani dal centro del video di 50 pixel, sia orizzontalmente che verticalmente. Per centrare sullo schermo uno sprite di dimensione doppia, la sua posizione orizzontale dovrebbe essere 160 e quella verticale 129 (figura 1.16). Per trovare le posizioni degli angoli sommate e sottraete 50 alla posizione centrale. La figura 2.15 mostra la posizione risultante.
Iniziate con lo sprite 0 nell'angolo in alto a sinistra, lo sprite 1 nell'angolo in basso a destra; lo sprite 0 deve muoversi verso il basso, poi a destra, verso l'alto e a sinistra. Lo sprite 1 deve muoversi verso l'alto, a sinistra, in basso e poi a destra. Guardate la figura 2.16: mostra quattro viste dei due sprite mentre si muovono lungo il percorso. La prima mostra le posizioni iniziali: le frecce indicano la direzione in cui si stanno muovendo gli sprite. Notate che quando uno sprite si muove verticalmente, anche l'altro si muove verticalmente, ma in direzione opposta. La stessa osservazione vale anche per i movimenti in orizzontale. Questa simme-

```
1000 REM *** INSEGUIMENTO ***
1550 POKE VIC,110     :REM POSIZ. ORIZZONTALE SPRITE 0
1560 POKE VIC+2,210   :REM POSIZ. ORIZZONTALE SPRITE 1
1570 POKE VIC+1,79    :REM POSIZ. VERTICALE SPRITE 0
1580 POKE VIC+3,179   :REM POSIZ. VERTICALE SPRITE 1
1662 :
1664 :
1666 REM ** INIZIALIZZA IL MOVIMENTO DEGLI SPRITE
1667 :
1668 D0 = 1 : D1 = -1
1670 :
1672 :
1674 REM ** SPOSTAMENTO IN VERTICALE
1676 :
1678 FOR MOV = 1 TO 100
1680 :    POKE VIC+1, PEEK(VIC+1) + D0
1682 :    POKE VIC+3, PEEK(VIC+3) + D1
1684 :    GET KP$
1686 :    IF KP$ = "" THEN 1690
1688 :       MOV = 100 : TASTO = -1
1690 NEXT MOV
1692 :
1694 :
1696 REM ** CONCLUDE, SE VIENE PREMUTO UN TASTO
1698 :
1700 IF TASTO THEN 1750
1702 :
1704 :
1706 REM ** SPOSTAMENTO IN ORIZZONTALE
1708 :
1710 FOR MOV = 1 TO 100
1712 :    POKE VIC, PEEK(VIC) + D0
1714 :    POKE VIC+2, PEEK(VIC+2) + D1
1716 :    GET KP$
1718 :    IF KP$ = "" THEN 1722
1720 :       MOV = 100 : TASTO = -1
1722 NEXT MOV
1724 :
1726 :
1728 REM ** CONCLUDE, SE VIENE PREMUTO UN TASTO
1730 :
1732 IF TASTO THEN 1750
1734 :
1736 :
1738 REM ** INVERTE IL MOVIMENTO E RIPETE
1740 :
1742 D0 = -D0 : D1 = -D1
1744 GOTO 1678
1746 :
1748 :
```

FIG. 2.14. Cambiamenti e aggiunte per cambiare il programma "Coppia di sprite" nel programma "Inseguimento".

tria di movimento fa in modo che la nostra programmazione sia più semplice.

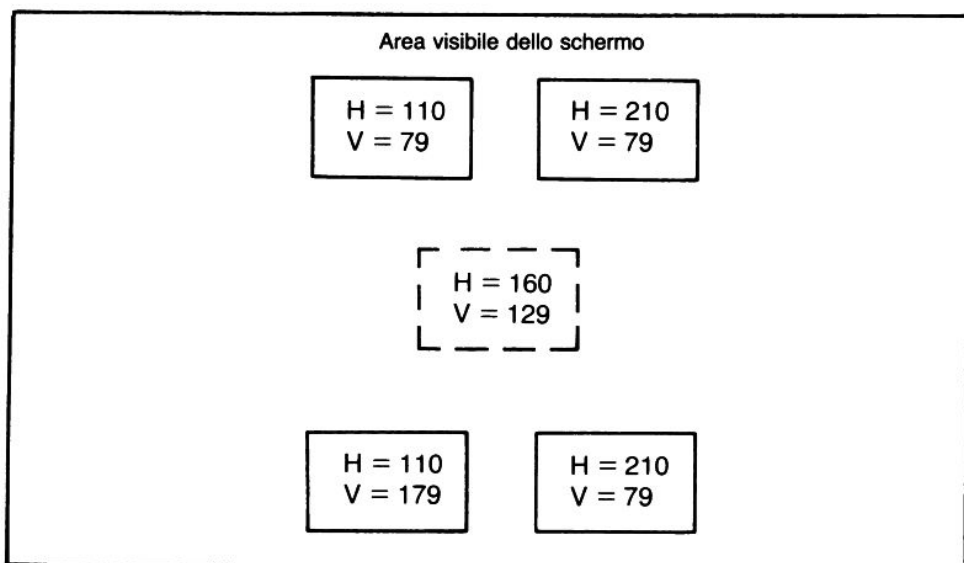



FIG. 2.15. Le posizioni ai vertici che sono raggiunte da uno sprite seguendo un percorso quadrato. Ogni lato del percorso è lungo 100 pixel e il percorso è centrato sullo schermo.

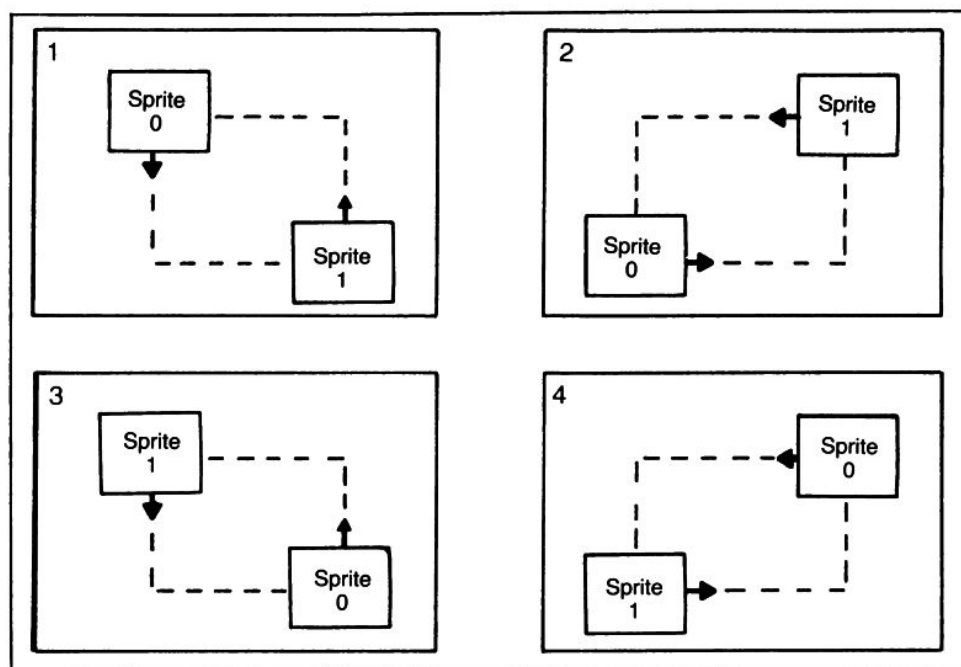



FIG. 2.16. Quattro viste dei due sprite mentre si muovono lungo il percorso quadrato.

Come si stabiliscono le posizioni e i movimenti degli sprite

Le istruzioni 1550-1580 fissano le posizioni iniziali degli sprite:

```
1550 POKE VIC,110   :REM POSIZ. ORIZZONTALE SPRITE 0
1560 POKE VIC+2,210 :REM POSIZ. ORIZZONTALE SPRITE 1
1570 POKE VIC+1,79  :REM POSIZ. VERTICALE SPRITE 0
1580 POKE VIC+3,179 :REM POSIZ. VERTICALE SPRITE 1
```

Come abbiamo detto prima, lo sprite 0 parte dall'angolo in alto a sinistra e lo sprite 1 parte dall'angolo a destra in basso. Il programma usa due variabili per produrre i movimenti dello sprite: D0 per lo sprite 0 e D1 per lo sprite 1. L'istruzione 1668 assegna a queste variabili il loro valore iniziale:

```
1668 D0 = 1 : D1 = -1
```

Sommeremo i valori di queste variabili di movimento ai registri di posizione degli sprite. Se sommiamo numeri positivi, il numero diventa maggiore e lo sprite si muoverà verso il basso. Sommando numeri negativi, le posizioni verticali avranno un valore più piccolo e lo sprite si muoverà verso l'alto. Per muoversi orizzontalmente si procede in modo analogo. Sommando numeri positivi alla posizione orizzontale si muoverà lo sprite verso destra e sommando numeri negativi lo si farà muovere a sinistra. Le istruzioni 1678-1690 riguardano gli spostamenti verticali lungo il percorso, per entrambi gli sprite:

```
1678 FOR MOV = 1 TO 100
1680 :   POKE VIC+1, PEEK(VIC+1) + D0
1682 :   POKE VIC+3, PEEK(VIC+3) + D1
1684 :   GET KP$
1686 :   IF KP$ = "" THEN 1690
1688 :      MOV = 100 : TASTO = -1
1690 NEXT MOV
```

Le istruzioni 1678 e 1690 inizializzano un ciclo che sarà eseguito 100 volte perché ogni lato del percorso è lungo 100 pixel: ci sposteremo di un pixel a ogni passaggio nel ciclo. Ogni volta che si è nel ciclo l'istruzione 1680 somma il valore della variabile di movimento dello sprite 0 al registro di posizione di quello sprite. Analogamente, la 1682 somma il valore della variabile di movimento dello sprite 1 alla sua posizione verticale.

Le istruzioni 1686-1688 rappresentano un miglioramento dei programmi per il movimento dello sprite. Ora possiamo far controllare al programma se è stato premuto un tasto dopo ogni movimento di uno

sprite, senza dover attendere la fine del ciclo. L'istruzione 1686 esplora la tastiera; se nessun tasto è stato digitato l'istruzione 1688 viene saltata e il ciclo continua il suo lavoro. Se è stato premuto un tasto accadono due cose: il valore della variabile di controllo del ciclo, MOV, aumenta fino a 100 e alla variabile TASTO (KEYPRESS, in altri programmi di questo libro) viene attribuito il valore − 1. Portando MOV a 100 verrà forzata un'uscita veloce del ciclo quando viene raggiunta l'istruzione 1690. TASTO è portata a − 1 perché − 1 rappresenta "vero". L'istruzione 1700 ci manderà o sul ciclo di movimento orizzontale o alla fine del programma, a seconda del valore di TASTO:

```
1700 IF TASTO THEN 1750
```

Se TASTO contiene uno 0, cioè rappresenta il falso, significa che nessun tasto è stato digitato e il programma entra nel ciclo di movimento orizzontale. Se TASTO contiene − 1, allora significa che un tasto è stato digitato; TASTO sarà quindi interpretato come "vero" e il programma andrà al segmento che comincia con l'istruzione 1750. Questi controlli vero/falso sono chiamati controlli booleani, e potete definire TASTO una variabile booleana. Le istruzioni 1710-1722 formano un ciclo che si occupa del movimento orizzontale lungo il percorso:

```
1710 FOR MOV = 1 TO 100
1712 :   POKE VIC, PEEK(VIC) + D0
1714 :   POKE VIC+2, PEEK(VIC+2) + D1
1716 :   GET KP$
1718 :   IF KP$ = "" THEN 1722
1720 :      MOV = 100 : TASTO = -1
```

Questo ciclo è quasi uguale a quello per il movimento verticale. L'unica differenza è che ora il programma sommerà i valori di movimento ai registri di posizione orizzontale. L'istruzione 1732 controlla se è stato digitato un tasto durante il ciclo precedente:

```
1732 IF TASTO THEN 1750
```

Se un tasto è stato digitato, il programma salterà all'istruzione 1750 e terminerà, altrimenti è ora di cambiare direzione.

Un piccolo trucco: come si cambia la direzione

Consideriamo lo sprite 0 in questo programma. Per completare il percorso, deve scendere, andare a destra, poi salire e infine andare a

sinistra. Potete pensarla anche in un altro modo: movimento verticale, movimento orizzontale, movimento verticale e movimento orizzontale. Abbiamo considerato due cicli che si preoccupano dei movimenti verticali e orizzontali. Secondo voi abbiamo bisogno di un'altra coppia di cicli? No, è sufficiente cambiare la direzione di movimento dello sprite 0 e poi ritornare agli stessi due cicli orizzontale e verticale. Il valore iniziale della variabile di movimento D0 era 1. Se lo moltiplichiamo per − 1, si ottiene − 1. Ora il ciclo di movimento verticale manderà lo sprite 0 verso l'alto, mentre quello orizzontale lo manderà verso sinistra. Allo stesso modo possiamo invertire le direzioni di movimento dello sprite 1. Il valore del suo movimento iniziale era − 1; moltiplicato per − 1, ci dà il valore di movimento 1. Ora scenderà nel ciclo di movimento verticale e andrà verso destra nel ciclo di movimento orizzontale. Quando entrambi i movimenti sono stati invertiti dovete ritornare all'istruzione 1678 e rientrare nei cicli di movimento. Le istruzioni 1742-1744 sono quelle che attuano questa inversione:

```
1742 D0 = -D0 : D1 = -D1
1744 GOTO 1678
```

Ogni volta che il programma riesegue l'istruzione 1742 i movimenti saranno effettuati ancora al contrario; questo li riporterà ai loro valori originari, e ciò è corretto perché a questo punto ogni sprite sarà tornato alla posizione iniziale: lo sprite 0 nell'angolo superiore sinistro e lo sprite 1 nell'angolo inferiore destro. Ora tocca a voi, divertitevi con l'"Inseguimento di sprite". Siete capaci di disegnare un percorso triangolare? O di muovere quattro sprite lungo un quadrato? O, ancora, di muovere gli sprite a spirale verso il centro e poi di farli tornare, sempre in un movimento a spirale, verso l'esterno? Ricordatevi di pensare prima e poi di scrivere le istruzioni dei programmi.

CHE COSA ABBIAMO IMPARATO

In questo capitolo abbiamo imparato qualche tecnica per trattare più di uno sprite alla volta, cioè:

– Come visualizzare più sprite usando per ciascuno gli stessi 63 byte di dati relativi ai pixel
– Il significato di bit e byte e come vengono utilizzati in alcuni registri del VIC-II per controllare i singoli sprite

– Come immagazzinare più di un blocco di dati di sprite e come inizializzare i puntatori di sprite a 2040-2047
– Una delle tecniche per ottenere il movimento di più di uno sprite in uno schema interessante
– Come si usano le variabili booleane per uscire dai cicli.

ESERCIZI

Ecco qualche esercizio per migliorare la vostra abilità.

Test

Le risposte sono date più avanti.

1. Di solito la locazione di memoria 2045 serve come puntatore ai dati dello sprite per lo sprite ....
2. Un gruppo di 8 bit è chiamato ....
3. Un byte può rappresentare numeri decimali compresi tra 0 e ....
4. Per visualizzare gli sprite 2, 4 e 7 si deve mettere il numero decimale .... nella locazione VIC + 21
5. Se si vuole che gli sprite 0, 3 e 4 si espandano verticalmente, si deve inserire il numero .... nella locazione VIC + 23.
6. Se si usano 8 blocchi di dati di sprite, una buona area di memoria per inserirli inizia nella locazione ....
7. Per inizializzare uno sprite di dimensione doppia a metà schermo, al suo registro di posizione verticale si deve attribuire il valore ....
8. Quando la posizione orizzontale dello sprite diventa maggiore, esso si muove verso il lato .... del video.
9. Le variabili che assumono valori che rappresentano il vero e il falso sono dette variabili ....
10. Per ottenere un valore di una variabile che oscilli tra – 1 e 1 moltiplichiamo ripetutamente la variabile per ....

Esercizi di programmazione

1. Cambiate il programma "Cloni semplici" in modo che appaiano altri quattro cloni, uno in ogni angolo del video.
2. Cambiate il programma "Inseguimento" in modo che la coppia si muova in senso orario.

3. Cambiate il programma "Inseguimento" in modo che due personaggi femminili rincorrano due personaggi maschili intorno a un quadrato.

Risposte al test

1. 5
2. byte
3. 255
4. 148
5. 25
6. 12288
7. 129
8. destra
9. booleane
10. − 1

Soluzioni possibili agli esercizi di programmazione

Queste soluzioni sono state ottenute aggiungendo o modificando istruzioni nei programmi citati negli esercizi.

1. Caricate il programma "Cloni semplici" e battete queste istruzioni:

```
1000 REM *** OTTO CLONI ***
1362 POKE 2044,14    :REM PUNTATORE DATI SPRITE 4
1364 POKE 2045,14    :REM PUNTATORE DATI SPRITE 5
1366 POKE 2046,14    :REM PUNTATORE DATI SPRITE 6
1368 POKE 2047,14    :REM PUNTATORE DATI SPRITE 7
1412 POKE VIC+8,24   :REM POSIZ. ORIZZONTALE SPRITE 4
1414 POKE VIC+10,64  :REM POSIZ. ORIZZONTALE SPRITE 5
1416 POKE VIC+12,24  :REM POSIZ. ORIZZONTALE SPRITE 6
1418 POKE VIC+14,64  :REM POSIZ. ORIZZONTALE SPRITE 7
1419 POKE VIC+16,160:REM 5&7 USANO SECONDO REGISTRO ORIZZ.
1462 POKE VIC+9,50   :REM POSIZ. VERTICALE SPRITE 4
1464 POKE VIC+11,50  :REM POSIZ. VERTICALE SPRITE 5
1466 POKE VIC+13,229:REM POSIZ. VERTICALE SPRITE 6
1468 POKE VIC+15,229:REM POSIZ. VERTICALE SPRITE 7
1512 POKE VIC+43,7   :REM SPRITE 4 GIALLO
1514 POKE VIC+44,5   :REM SPRITE 5 VERDE
1516 POKE VIC+45,3   :REM SPRITE 6 CYAN
1518 POKE VIC+46,1   :REM SPRITE 7 BIANCO
1530 POKE VIC+21,255:REM SPRITE 0-7 ATTIVATI
```

2. Caricate il programma "Inseguimento" e inserite queste istruzioni:

```
1000 REM *** IN SENSO ORARIO ***
1674 REM ** SPOSTAMENTO ORIZZONTALE
```

```
1680 :    POKE VIC, PEEK(VIC) + D0
1682 :    POKE VIC+2, PEEK(VIC+2) + D1
1706 REM ** SPOSTAMENTO VERTICALE
1712 :    POKE VIC+1, PEEK(VIC+1) + D0
1714 :    POKE VIC+3, PEEK(VIC+3) + D1
```

3. Caricate il programma "Inseguimento" e inserite queste istruzioni:

```
1000 REM *** INSEGUICOPPIE ***
1530 POKE 2041,14    :REM PUNTATORE DATI SPRITE 1
1533 POKE 2042,15    :REM PUNTATORE DATI SPRITE 2
1536 POKE 2043,15    :REM PUNTATORE DATI SPRITE 3
1563 POKE VIC+4,110  :REM POSIZ. ORIZZONTALE SPRITE 2
1566 POKE VIC+6,210  :REM POSIZ. ORIZZONTALE SPRITE 3
1568 :
1583 POKE VIC+5,179  :REM POSIZ. VERTICALE SPRITE 2
1586 POKE VIC+7,79   :REM POSIZ. VERTICALE SPRITE 3
1613 POKE VIC+41,1   :REM SPRITE 2 BIANCO
1616 POKE VIC+42,5   :REM SPRITE 3 VERDE
1630 POKE VIC+23,15  :REM TUTTI I QUATTRO SPRITE
1640 POKE VIC+29,15  :REM IN DIMENSIONE DOPPIA
1660 POKE VIC+21,15  :REM TUTTI E QUATTRO ATTIVATI
1674 REM ** SU UN LATO DEL PERCORSO
1681 :    POKE VIC+4, PEEK(VIC+4) + D0
1683 :    POKE VIC+6, PEEK(VIC+6) + D1
1706 REM ** ALTRO LATO DEL PERCORSO
1713 :    POKE VIC+5, PEEK(VIC+5) + D1
1715 :    POKE VIC+7, PEEK(VIC+7) + D0
```

CAPITOLO TRE

Altri trucchi per gli sprite

Vi piacerebbe visualizzare sprite disegnati con più colori? Questo capitolo vi mostrerà come fare.
Imparerete inoltre a fare scorrere gli sprite l'uno sopra o sotto l'altro. Infine userete un'insieme di immagini di sprite per creare alcune divertenti animazioni. Vi farete ancora un po' di esperienza su bit, byte e movimenti di sprite.

SPRITE MULTICOLORI

Un disegno di uno sprite normale è definito immagazzinando nella memoria del computer 63 byte di informazioni sui pixel.
Ogni byte contiene 8 bit, e ogni bit accende o spegne un pixel. Dato che 63 × 8 è uguale a 504, si possono definire sprite che contengono 504 pixel.
Se un bit di un pixel ha valore 1, quel pixel si colorerà del colore che avete memorizzato nel registro del colore dello sprite. Se un bit di pixel è 0, quel pixel assumerà il colore dello schermo; cioè non si noterà.
Dato che normalmente esiste un unico pixel su cui operare, un pixel ha due possibilità: essere evidente o invisibile.
Comunque il Commodore ci offre un'altra alternativa: il modo sprite multicolore. In questo modo potete usare due bit (una "coppia di bit") per scegliere un colore.
Due bit possono contenere quattro possibili configurazioni di bit, come mostrato nella figura 3.1. E ciò vuol dire che potete scegliere uno

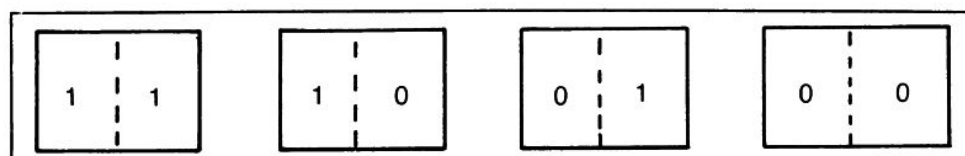



FIG. 3.1. Due bit possono contenere 4 possibili configurazioni di bit.

qualsiasi fra quattro colori per i due punti controllati da una coppia di bit. Ovviamente, i due punti avranno lo stesso colore.
È meglio pensare questi due punti come un pixel di larghezza doppia. Così ogni riga dell'immagine di uno sprite avrà 12 pixel di larghezza doppia, invece di 24 pixel di dimensione normale. Lo sprite avrà così più colore, ma meno dettagli orizzontali.
Poiché sono necessari due bit per scegliere un colore, ogni byte controlla soltanto 4 pixel di larghezza doppia.
Guardate la figura 3.2: il disegno dello sprite contiene 3 bit per riga, ciò significa che lo sprite sarà di 12 pixel di dimensione doppia di larghezza. Esso verrà visualizzato con la stessa dimensione di uno sprite normale, ma con meno dettagli orizzontali. Dato che usiamo ancora 63 byte per definire il disegno dello sprite, esso sarà composto da 252 pixel di dimensione doppia (63 × 4).

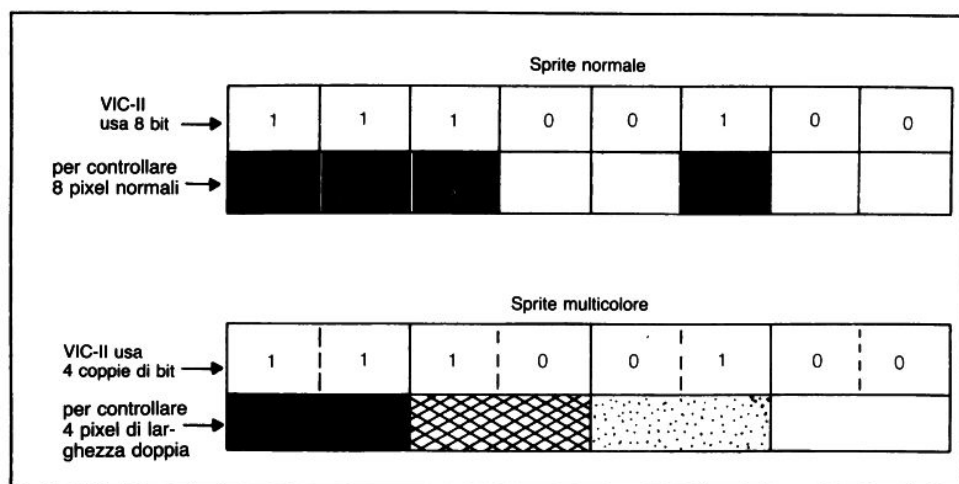



FIG. 3.2. In uno sprite multicolore ogni coppia di bit controlla il colore di un pixel di larghezza doppia.

ANCORA SUL MODO MULTICOLORE

Che colori assumerà uno sprite multicolore? Se la coppia di bit è 00, al pixel di larghezza doppia sarà attribuito il colore del video. Incidentalmente, il colore dello schermo è controllato dal numero nel registro VIC + 33 (53281).
Se la coppia di bit è 01, il colore deriverà dal registro 0 dello sprite multicolore in VIC + 37.
Se la coppia di bit è 10, il pixel prenderà il suo colore dal normale registro del colore dello sprite. Ricordate: ogni sprite ha il proprio registro del colore in una delle locazioni tra VIC + 39 e VIC + 46. Se la coppia di bit è 11, il colore sarà attribuito dal registro multicolore dello sprite 1 in VIC + 38. La figura 3.3 riassume tutto quello che abbiamo detto.
Come trasmettere al Commodore 64 che volete stampare uno sprite nel modo multicolore? Il registro VIC + 28 è un selezionatore di sprite multicolori. Ogni bit controlla uno sprite nel solito modo: il bit 0 controlla lo sprite 1, e via di seguito. Ponendo a 1 un bit dello sprite in VIC + 28, trasformerete quello sprite in modo multicolore.

DISEGNARE UNO SPRITE MULTICOLORE

Qualcuno di voi saprà forse disegnare uno sprite multicolore mentalmente, ma quasi tutti hanno bisogno di un po' d'aiuto.
La figura 3.4 rappresenta una scheda di codificazione per sprite multico-

Coppia di bit	Descrizione	Locazione
0 0	Colore dello schermo	VIC+33 (53281)
0 1	Registro di sprite multicolore 0	VIC+37 (53285)
1 0	Registro di colore dello sprite	Uno dei registri VIC+39– VIC+46 (53287-53294)
1 1	Registro dello sprite multicolore 1	VIC+38 (53286)

FIG. 3.3. In uno sprite multicolore ogni paio di bit prende il proprio colore da un particolare registro di VIC.

Colonna numero	0		1		2		3		4		5		6		7		8		9		10		11		Numero codici		
Valori	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1			
Riga 0																											
Riga 1																											
Riga 2																											
Riga 3																											
Riga 4																											
Riga 5																											
Riga 6																											
Riga 7																											
Riga 8																											
Riga 9																											
Riga 10																											
Riga 11																											
Riga 12																											
Riga 13																											
Riga 14																											
Riga 15																											
Riga 16																											
Riga 17																											
Riga 18																											
Riga 19																											
Riga 20																											

Colore dello schermo trasparente	0	0	Registro multicolore 0	0	1	Colore dello sprite	1	0	Registro multicolore 1	1	1

FIG. 3.4. Una scheda speciale codificata per gli sprite multicolori.

lori. È molto simile alla scheda per gli sprite normali mostrata nella figura 1.6.

Ci sono ancora 21 righe, valori di bit in ogni posizione e sulla destra tre colonne con la codifica. Comunque, ci sono solo 12 colonne, dato che i nostri pixel sono di larghezza doppia.

Come si usa questa scheda? Facciamo riferimento alla figura 3.5, che mostra una scheda di codifica di uno sprite multicolore già riempita. Prima di tutto introducete in fondo alla scheda i rettangolini colorati. Date a ogni colore una sfumatura diversa. Di solito è più semplice lasciare che il colore dello schermo sia il bianco.

Poi colorate i pixel di doppia larghezza usando le sfumature che avete segnato nei rettangolini in basso. Quando otterrete un disegno che sia di vostro gradimento, introducete gli 1; non preoccupatevi degli 0. Usando i rettangolini colorati in basso come una guida, introducete tutte le posizioni dei bit che devono contenere un 1. Troverete più semplice fare tutti i pixel di un colore prima di procedere col colore successivo.

Infine, è ora di sommare i valori dei bit.

Colonna numero	0	1	2	3	4	5	6	7	8	9	10	11	Numero codici		
Valori	128 64	32 16	8 4	2 1	128 64	32 16	8 4	2 1	128 64	32 16	8 4	2 1			
Riga 0													1	85	64
Riga 1													1	85	64
Riga 2													1	20	64
Riga 3													1	20	64
Riga 4													1	85	64
Riga 5													1	20	64
Riga 6													1	65	64
Riga 7													1	85	64
Riga 8													0	60	0
Riga 9													0	60	0
Riga 10													62	170	188
Riga 11													62	170	188
Riga 12													48	170	12
Riga 13													16	170	4
Riga 14													20	130	20
Riga 15													20	130	20
Riga 16													16	195	4
Riga 17													0	195	0
Riga 18													0	65	0
Riga 19													1	65	64
Riga 20													1	65	64

Colore dello schermo trasparente 0 0 — Registro multicolore 0 0 1 — Colore dello sprite 1 0 — Registro multicolore 1 1 1

FIG. 3.5. Esempio di una scheda compilata per uno sprite multicolore.

Per ogni byte, sommate tutti i valori dei bit che contengono un 1. La somma viene introdotta nell'opportuno rettangolo del codice numerico alla destra della scheda.
Riguardate la figura 3.5. Accertatevi di aver capito come abbiamo ottenuto i 63 numeri di codifica. Poi fate alcune copie della scheda e introducete il vostro disegno. Lo userete nel prossimo paragrafo.

UN PROGRAMMA PER STAMPARE GLI SPRITE A COLORI

La figura 3.6 è un listato del programma "Sprite 4 colori". Il programma porta sullo schermo i caratteri disegnati in figura 3.5. Digitando un tasto, il programma termina.
Questo programma è molto simile ai nostri primi programmi di stampa di uno sprite. La grande differenza sta nelle istruzioni **1400-1440**.

```
1000 REM *** SPRITE 4 COLORI ***
1010 :
1020 :
1030 REM ** PROVVEDE UN FEEDBACK SULLO SCHERMO
1040 :
1050 PRINT "[illegible]STO PENSANDO ";
1060 :
1070 :
1080 REM ** CARICA I DATI DEGLI SPRITE
1090 :
1100 FOR N = 896 TO 958
1110 :   READ SPDTA
1120 :   POKE N, SPDTA
1130 NEXT N
1140 :
1150 DATA   1,  85,  64,   1,  85,  64
1160 DATA   1,  20,  64,   1,  20,  64
1170 DATA   1,  85,  64,   1,  20,  64
1180 DATA   1,  65,  64,   1,  85,  64
1190 DATA   0,  60,   0,   0,  60,   0
1200 DATA  62, 170, 188,  62, 170, 188
1210 DATA  48, 170,  12,  16, 170,   4
1220 DATA  20, 130,  20,  20, 130,  20
1230 DATA  16, 195,   4,   0, 195,   0
1240 DATA   0,  65,   0,   1,  65,  64
1250 DATA   1,  65,  64
1260 :
1270 :
1280 REM ** FISSA I CONTROLLI DEGLI SPRITE
1290 :
1300 PRINT "[illegible]";
1310 POKE 2040,14    :REM PUNTA AI DATI
1320 VIC = 53248     :REM CHIP GRAFICO
1330 :
1340 POKE VIC,160    :REM POS ORIZZONTALE
1350 POKE VIC+1,129  :REM POS VERTICALE
1360 :
1370 POKE VIC+23,1   :REM ESPANSIONE VERTICALE
1380 POKE VIC+29,1   :REM ESPANSIONE ORIZZONTALE
1390 :
1400 POKE VIC+28,1   :REM MULTICOLORE 0
1410 POKE VIC+33,0   :REM SFONDO NERO
1420 POKE VIC+37,7   :REM MCR 0 GIALLO
1430 POKE VIC+39,5   :REM SPR 0 VERDE
1440 POKE VIC+38,6   :REM MCR 1 BLU
1450 :
1460 POKE VIC+21,1   :REM SPRITE 0 ATTIVO
1470 :
1480 :
1490 REM ** ASPETTA CHE VENGA PREMUTO UN TASTO
1500 :
1510 GET KP$
1520 IF KP$ = "" THEN 1510
1530 :
1540 :
1550 REM ** REINIZIALIZZA I CONTROLLI DEGLI SPRITE
1560 :
1570 POKE VIC+21,0   :REM SPRITE DISATTIVATI
1580 POKE VIC+28,0   :REM MULTICOLORE DISATTIVATO
1590 POKE VIC+29,0   :REM ESPANS. ORIZZ. DISATTIVATA
1600 POKE VIC+23,0   :REM ESPANS. VERT.  DISATTIVATA
1610 :
1620 END
```

FIG. 3.6. Listato del programma "Sprite 4 colori".

```
1400 POKE VIC+28,1   :REM MULTICOLORE 0
1410 POKE VIC+33,0   :REM SFONDO NERO
1420 POKE VIC+37,7   :REM MCR 0 GIALLO
1430 POKE VIC+39,5   :REM SPR 0 VERDE
1440 POKE VIC+38,6   :REM MCR 1 BLU
```

L'istruzione 1400 inizializza il registro di selezione dello sprite multicolore così lo sprite 0 verrà stampato multicolore.
Le istruzioni 1410-1440 inizializzano poi i 4 colori che verranno usati: nero, indicato dalla coppia di bit 00, giallo 01, verde 10, infine blu 11.
Esiste un'altra differenza: alla fine del programma, dovete reinizializzare il registro di selezione multicolore (l'istruzione 1580).
Uno sprite disegnato per una stampa normale appare abbastanza strano, se visualizzato in modo multicolore. Se volete verificare quanto strano, ritornate a qualcuno dei nostri primi programmi, e inserite le istruzioni 1410-1440 che determinano il modo multicolore.
Inserite il programma "Sprite 4 colori", salvatelo e fatelo girare. Divertitevi con le diverse possibilità di colore elencate nelle istruzioni 1410-1440 per vedere se ottenete una combinazione più piacevole.
Quando avete terminato gli esperimenti, è il momento di provare a introdurre una vostra codifica.
Reinserite i dati dei pixel nelle istruzioni 1150-1250 di "Sprite 4 colori" con i numeri di codifica che avevate ottenuto nell'ultimo paragrafo. Poi rifate girare il programma.
Come vi pare? Occorre un po' d'ingegno per poter ottenere il risultato voluto.

SOPRA E SOTTO

Quando uno sprite si muove lungo lo schermo, può coprire una parte dell'area usata da un altro sprite.
Quando ciò succede, una priorità prefissata determina quale sprite ha la precedenza rispetto a un altro sprite.
Lo sprite 0 ha la priorità massima e lo sprite 7 ha la priorità minima. Così, se lo sprite 0 divide parte della stampa con lo sprite 7, ha la precedenza.
Allo stesso modo, lo sprite 4 ha la precedenza sullo sprite 5. La figura 3.7 riassume queste priorità.
Se uno sprite è davanti a un altro sprite è possibile vedere parti dello sprite nascosto: le parti dello sprite con priorità maggiore che sono trasparenti, cioè i pixel che hanno il colore dello schermo, agiranno come una finestra. Potrete vedere attraverso questa finestra parti dello sprite con priorità minore.

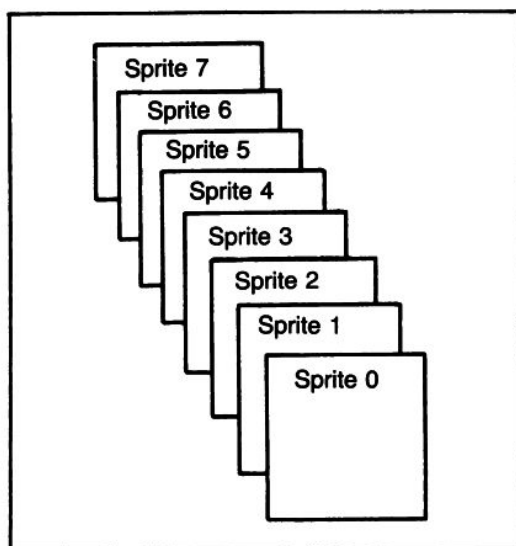



FIG. 3.7. Quando gli sprite si incontrano, la priorità superiore va agli sprite con i numeri più bassi, i quali restano visibili davanti agli sprite con i numeri più alti.

La figura 3.8 presenta un listato del programma "Sovrapposizione di sprite". Inserite il programma nel computer, salvatelo, e poi fatelo girare. Guardatelo per un po'.
"Sovrapposizione di sprite" visualizza sullo schermo quattro sprite simili, che cominciano un'interminabile danza in quadrato, fino a quando non si digita un tasto.
Notate come le parti trasparenti degli sprite 1 e 0 lascino vedere le parti degli sprite su cui passano.
Questo programma mostra due caratteristiche interessanti: la prima è il modo con cui sono definite le forme dello sprite; la seconda è il modo in cui è inizializzata la danza in quadrato.

Cicli che generano sprite

Le istruzioni 1100-1120 costituiscono il blocco dei dati relativi ai pixel dello sprite per il programma "Sovrapposizione":

```
1100 FOR N = 832 TO 894
1110 :   POKE N, 60
1120 NEXT N
```

```
1000 REM *** SOVRAPPOSIZIONE ***
1010 :
1020 :
1030 REM ** PROVVEDE UN FEEDBACK SULLO SCHERMO
1040 :
1050 PRINT "♥QQQQQQQQQMI STO PREPARANDO";
1060 :
1070 :
1080 REM ** CARICA I DATI DEGLI SPRITE
1090 :
1100 FOR N = 832 TO 894
1110 :     POKE N, 60
1120 NEXT N
1130 :
1140 :
1150 REM ** FISSA I CONTROLLI DEGLI SPRITE
1160 :
1170 PRINT "♥"        :REM PULISCE LO SCHERMO
1180 VIC = 53248      :REM CHIP GRAFICO
1190 :
1200 POKE 2040,13     :REM PUNTATORE DATI SPRITE 0
1210 POKE 2041,13     :REM PUNTATORE DATI SPRITE 1
1220 POKE 2042,13     :REM PUNTATORE DATI SPRITE 2
1230 POKE 2043,13     :REM PUNTATORE DATI SPRITE 3
1240 :
1250 POKE VIC,226     :REM POS ORIZZONTALE SPRITE 0
1260 POKE VIC+2,94    :REM POS ORIZZONTALE SPRITE 1
1270 POKE VIC+4,144   :REM POS ORIZZONTALE SPRITE 2
1280 POKE VIC+6,176   :REM POS ORIZZONTALE SPRITE 3
1290 :
1300 POKE VIC+1,140   :REM POS VERTICALE SPRITE 0
1310 POKE VIC+3,118   :REM POS VERTICALE SPRITE 1
1320 POKE VIC+5,190   :REM POS VERTICALE SPRITE 2
1330 POKE VIC+7,68    :REM POS VERTICALE SPRITE 3
1340 :
1350 POKE VIC+39,7    :REM 0 GIALLO
1360 POKE VIC+40,5    :REM 1 VERDE
1370 POKE VIC+41,3    :REM 2 CYAN
1380 POKE VIÇ+42,1    :REM 3 BIANCO
1390 :
1400 POKE VIC+23,15   :REM TUTTI GLI SPRITE
1410 POKE VIC+29,15   :REM IN DOPPIA DIMENSIONE
1420 :
1430 POKE VIC+21,15   :REM SPRITE 0-3 ATTIVATI
1440 :
1450 :
1460 REM ** REGOLA I REGISTRI DEL MOVIMENTO
1470 REM     E LE MOSSE INIZIALI
1480 :
1490 MR(0) = VIC   : MR(1) = VIC+2
1500 MR(2) = VIC+5 : MR(3) = VIC+7
1510 :
1520 MV(0) = -1 : MV(1) = 1
1530 MV(2) = -1 : MV(3) = 1
1540 :
1550 DF = -1 :REM -1:INTERNO, 0:ESTERNO
1560 :
1570 :
1580 REM ** SPOSTA GLI SPRITE
1590 :
1600 FOR COUNT = 1 TO 200
1610 :     SPRNUM = INT((COUNT-1)/50)
1620 :     IF DF THEN SPRNUM = 3 - SPRNUM
1630 :     REG = MR(SPRNUM)
```

```
1640 :   MOVE = MV(SPRNUM)
1650 :   POKE REG, PEEK(REG) + MOVE
1660 :   GET KP$
1670 :   IF KP$ = "" THEN 1690
1680 :      COUNT = 280 : KEYPRESS = -1
1690 NEXT COUNT
1700 :
1710 :
1720 REM ** TERMINA SE E' STATO PREMUTO UN TASTO
1730 :
1740 IF KEYPRESS THEN 1900
1750 :
1760 :
1770 REM ** PAUSA, POI INVERTE LA
1780 REM    DIREZIONE E RIPETE
1790 :
1800 FOR DELAY = 1 TO 400 : NEXT DELAY
1810 FOR SPRNUM = 0 TO 3
1820 :   MV(SPRNUM) = -1 * MV(SPNUM)
1830 NEXT SPRNUM
1840 DF = -1 - DF
1850 GOTO 1600
1860 :
1870 :
1880 REM ** CONCLUDE REINIZIALIZZANDO
1890 :
1900 POKE VIC+21,0
1910 POKE VIC+29,0
1920 POKE VIC+23,0
1930 :
1940 END
```

FIG. 3.8. Listato del programma “Sovrapposizione”.

Vi ricorderete di aver usato una tecnica simile nel primo programma “Uno sprite semplice”. In quel caso avevate assegnato il numero 255, che illuminava tutti i pixel nello sprite. In questo caso, abbiamo scelto il numero 60, che illumina i quattro pixel centrali di ogni gruppo di otto. Guardate la figura 3.9. Con tre figure del genere su ogni riga, uscirà il disegno di uno sprite costituito da 3 strisce verticali.
Potrete crearvi altri esempi cambiando questo ciclo. Inserite nel programma due nuove linee:

```
1100 FOR N = 832 TO 894 STEP 2
1110 :   POKE N, 255 : POKE N+1, 0
```

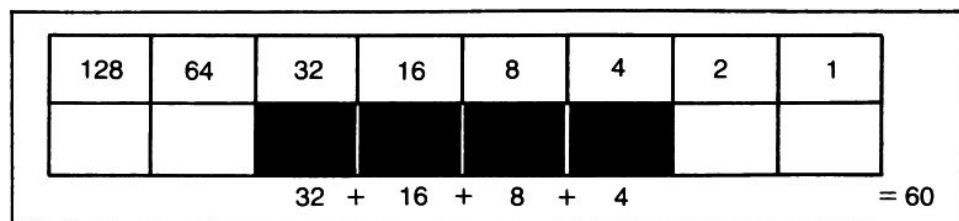



FIG. 3.9. L'inserimento di 60 come dato dello sprite illumina i 4 pixel di mezzo in ogni gruppo di 8.

Fate girare la nuova versione del programma.
È un puzzle interessante vedere quanti sprite complessi si possono disegnare solo attraverso un uso intelligente dei cicli.

Una danza su un percorso quadrato

All'inizio del movimento in "Sovrapposizione", i quattro sprite si trovano nelle posizioni mostrate in figura 3.10.
Uno alla volta gli sprite si muoveranno verso il centro dello schermo. Quando sono tutti raggruppati in quel punto, si fermano e poi ritornano alle loro posizioni d'origine uno alla volta. Dopo una breve pausa il movimento si ripeterà.
Quando si deve programmare un movimento, è utile cercare modelli che si ripetono. Questi modelli possono semplificare la programmazione. In questo caso, ogni sprite deve eseguire le stesse azioni: muoversi verso l'interno e poi verso l'esterno. Potete utilizzare un segmento di programma che attribuisce questi movimenti per uno sprite e poi cambiare lo sprite con cui lavora. Se inizializzate le variabili di movimento come matrici, sarà facile commutare gli sprite, facendo variare gli indici della matrice nell'istruzione di movimento.
Esiste un'altra utile semplificazione. Il movimento verso l'interno o verso l'esterno differiranno solo nella direzione presa dallo sprite. Tutto ciò che si deve fare è capovolgere la direzione dello sprite attraverso ripetizioni delle istruzioni di movimento. Così lo stesso segmento di programma farà muovere i quattro sprite, sia verso l'interno, sia verso l'esterno.
Ora dobbiamo elaborare solo i dettagli (le ultime parole famose di molti programmatori).

Inizializzazione dei registri e dei movimenti

Poiché ciascuno sprite sarà spostato solo verticalmente od orizzontalmente, ci sarà bisogno di un unico registro di posizione per poter muovere quello sprite. Le istruzioni 1490-1500 inizializzano i 4 registri che saranno usati per i movimenti dello sprite:

```
1490 MR(0) = VIC   : MR(1) =.VIC+2
1500 MR(2) = VIC+5 : MR(3) = VIC+7
```

Riguardate la figura 3.10. Gli sprite 0 e 1 si muoveranno orizzontalmente, gli sprite 2 e 3 si muoveranno invece verticalmente.

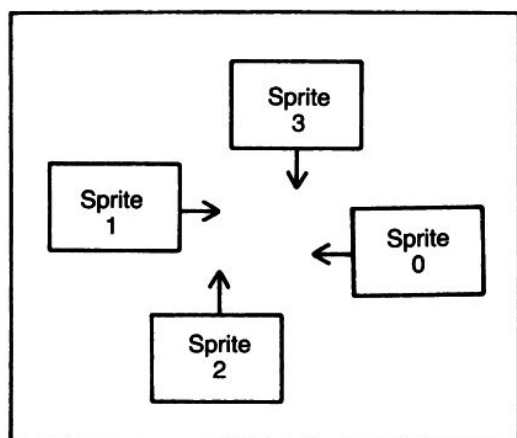



FIG. 3.10. Posizioni iniziali di 4 sprite nel programma "Sovrapposizione"; le frecce indicano la direzione del loro movimento iniziale.

Ho usato queste informazioni per poter stabilire quali registri di posizione usare.
Le istruzioni 1520-1530 danno a ogni sprite un movimento iniziale:

```
1520 MV(0) = -1 : MV(1) = 1
1530 MV(2) = -1 : MV(3) = 1
```

Il valore di questa variabile di movimento deve essere sommato alla posizione corrente dello sprite per attribuirgli un'altra posizione. Per controllare la logica di questi assegnamenti, riosservate la figura 3.10. Le frecce indicano la direzione del movimento iniziale di ogni sprite. Per esempio, lo sprite 3 comincerà con il muoversi verso il basso. Ogni volta che si muove, il valore della sua posizione verticale aumenta; è quello che fa l'istruzione di movimento associata allo sprite 3 dalla linea 1530.
Quando arriva il momento che lo sprite 3 inverta il proprio movimento, si deve moltiplicare il valore di MV(3) per (– 1). Poi la posizione verticale dello sprite diminuirà di 1 ogni volta che lo sprite si muoverà verso l'alto.
Bisogna però considerare un altro particolare: l'ordine con cui gli sprite si muoveranno. Quando il movimento è verso l'interno, l'ordine di spostamento degli sprite deve essere: 3, 2, 1, 0.
Quando gli sprite si muovono verso l'esterno, vogliamo muovere prima 0, seguito da 1, 2 e 3. Semplicemente si inverte l'ordine.
L'istruzione 1550 inizializza una variabile che tiene traccia dei movi-

menti verso l'interno e l'esterno, così avremo l'ordine corretto dei movimenti dello sprite:

```
1600 FOR COUNT = 1 TO 200
```

Il ciclo generale di movimento

Le istruzioni 1600-1690 muovono gli sprite:

```
1600 FOR COUNT = 1 TO 200
1610 :    SPRNUM = INT((COUNT-1)/50)
1620 :    IF DF THEN SPRNUM = 3 - SPRNUM
1630 :    REG = MR(SPRNUM)
1640 :    MOVE = MV(SPRNUM)
1650 :    POKE REG, PEEK(REG) + MOVE
1660 :    GET KP$
1670 :    IF KP$ = "" THEN 1690
1680 :       COUNT = 200 : KEYPRESS = -1
1690 NEXT COUNT
```

Le istruzioni 1600-1690 inizializzano un ciclo che si ripeterà 200 volte, a meno che venga premuto un tasto per terminare il programma.
Ogni sprite si muoverà 50 volte, un pixel alla volta, e ci sono 4 sprite da muovere: 4 × 50 = 200.
Le istruzioni 1610 e 1620 indicano quale sprite debba essere mosso, e assegnano il suo numero alla variabile SPRNUM.
Se gli sprite si muovono internamente, DF avrà il valore − 1.
SPRNUM acquisterà i valori 3, 2, 1 e poi 0, al procedere del ciclo.
Se gli sprite si muovono verso l'esterno, DF avrà il valore 0.
Ora SPRNUM acquisterà i valori 0, 1, 2 e poi 3, come volevamo.
L'istruzione 1630 prende il registro di posizione per regolarlo, sulla base della variabile SPRNUM, e l'istruzione 1640 seleziona il movimento dello sprite. L'istruzione 1650 esegue il lavoro effettivo, prende la vecchia posizione dello sprite selezionato e somma ad essa il movimento appropriato.
L'istruzione 1660 controlla se viene premuto qualche tasto. In caso positivo, l'istruzione 1670 provvede all'uscita immediata dal programma.
Se gli sprite si sono mossi verso l'interno, vogliamo ora farli muovere verso l'esterno. E se sono andati verso l'esterno vogliamo averli pronti per tornare verso l'interno. Le istruzioni 1810-1850 preparano al prossimo giro di danza:

```
1810 FOR SPRNUM = 0 TO 3
1820 :   MV(SPRNUM) = -1 * MV(SPNUM)
1830 NEXT SPRNUM
1840 DF = -1 - DF
1850 GOTO 1600
```

In primo luogo, il movimento degli sprite può essere invertito moltiplicando per − 1. Poi la variabile di movimento interno/esterno è commutata nell'istruzione 1840. Se era − 1, diventa 0; e se era 0 diventa − 1. Poi l'istruzione 1850 riconduce il programma al ciclo principale della danza, che inizia alla linea 1600.
Il programma girerà con il movimento degli sprite interno ed esterno, finché non verrà premuto un tasto o verrà staccata la spina.
Questa è una grande occasione per potervi divertire un po' con i cicli di movimento. Facendo delle modifiche al programma "Sovrapposizione di sprite", otterrete altre danze di sprite. Qui sono elencate alcune idee.
Fate in modo che due sprite si muovano contemporaneamente.
Fate in modo che gli sprite si spostino a nuove posizioni iniziali, quando si muovono verso l'esterno.
Fate in modo che ogni sprite ne copra un altro completamente, quando si sovrappongono.

COSTRUIRE CARTONI ANIMATI

L'animazione è una grande forma di magia. Mostrando velocemente una serie di immagini statiche, potete creare l'illusione del movimento e della vita. Fino ad ora i nostri sprite hanno avuto un'animazione molto limitata. Un'immagine si muove lungo lo schermo, ma non cambia la sua forma.
Ora proveremo alcune animazioni in cui l'immagine cambia: è facile, con gli sprite. Iniziamo caricando varie immagini di sprite. Poi stabiliremo un ciclo per un puntatore ai dati dello sprite che indichi a turno le immagini.

Sviluppare le immagini

Fissiamo uno di questi cicli di animazione.
La figura 3.11 mostra 3 immagini di un giocoliere. Notate come l'azione progredisca da immagine a immagine e come l'ultima immagine riporti alla prima. Organizzare un ciclo di immagini richiede un po' di riflessione e di lavoro.
Io di solito arrivo con un insieme preliminare di immagini e poi faccio girare un programma per poterle visualizzare. Poi modifico i dati finché ottengo l'effetto voluto. Le idee per le aggiunte e i cambiamenti all'animazione vengono spontaneamente provate, poi conservate o scartate. Le immagini di figura 3.11 sono i risultati finali di questo processo.

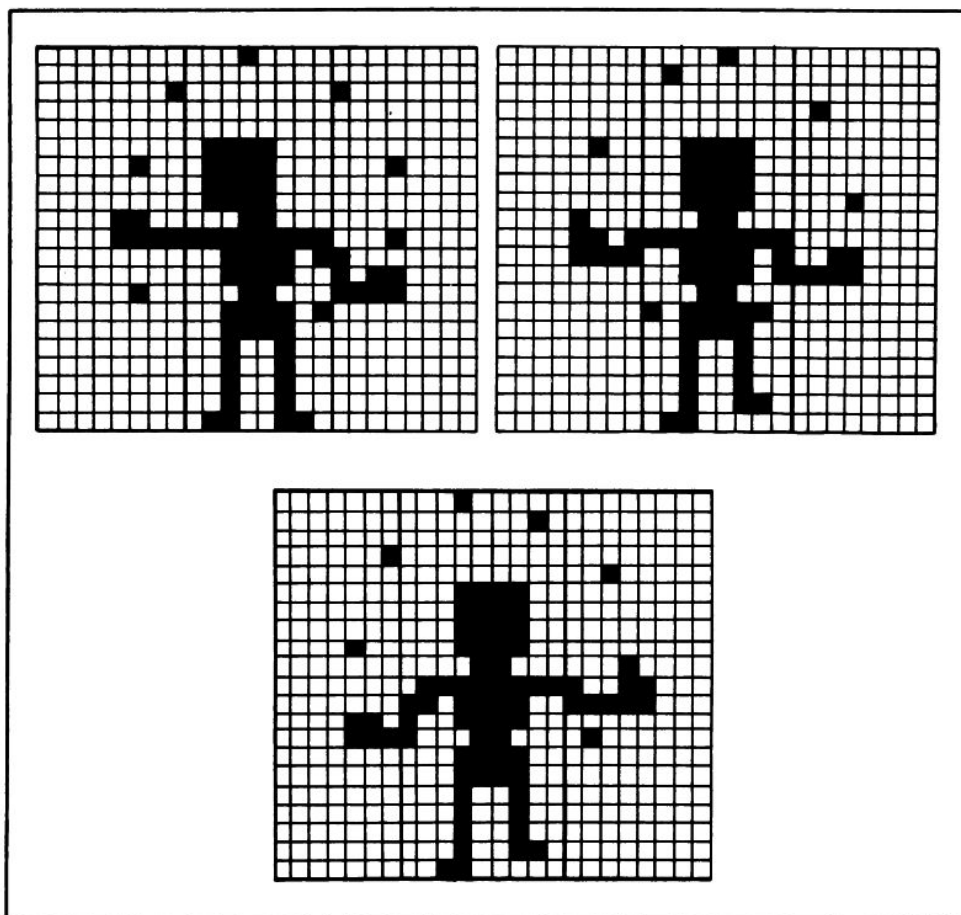



FIG. 3.11. Le tre immagini di sprite usate per animare un giocoliere nel programma "Giocoliere pazzo".

Una volta sviluppate le immagini, potete usare il ciclo di animazione in programmi diversi. Dopo un po', potete sviluppare un'intera biblioteca di questi insiemi di immagini animate.
Ora tocca a voi.
Usando la scheda di codifica degli sprite, sviluppate un insieme preliminare di 3 immagini, che formino un ciclo di animazione. L'azione in ogni immagine deve portare alla successiva e l'ultima deve portare alla prima.
Se siete a corto di idee, ecco qualche suggerimento per cicli semplici:
- una palla che rimbalza;

```
1000 REM *** GIOCOLIERE PAZZO ***
1010 :
1020 :
1030 REM ** PROVVEDE UN FEEDBACK SULLO SCHERMO
1040 :
1050 PRINT "]QQQQQQQQQQMI STO PREPARANDO";
1060 :
1070 :
1080 REM ** CARICA I DATI DEGLI SPRITE
1090 :
1100 FOR N = 832 TO 1023
1110 :    READ SPDTA
1120 :    IF SPDTA = -1 THEN PRINT " ."; : GOTO 1140
1130 :    POKE N, SPDTA
1140 NEXT N
1150 :
1160 DATA   0,  16,   0,   0,   0,   0
1170 DATA   1,   0, 128,   0,   0,   0
1180 DATA   0,   0,   0,   0, 120,   0
1190 DATA   4, 120,  16,   0, 120,   0
1200 DATA   0, 120,   0,  12,  24,   0
1210 DATA  15, 255,  16,   0,  61, 128
1220 DATA   0,  60, 176,   4,  24, 240
1230 DATA   0,  61,   0,   0,  60,   0
1240 DATA   0,  36,   0,   0,  36,   0
1250 DATA   0,  36,   0,   0,  36,   0
1260 DATA   0, 102,   0,  -1
1270 :
1280 DATA   0,   8,   0,   0,  64,   0
1290 DATA   0,   0,   0,   0,   0,  64
1300 DATA   0,   0,   0,   4,  60,   0
1310 DATA   0,  60,   0,   0,  60,   0
1320 DATA   0,  60,  16,   8,  24,   0
1330 DATA  13, 255,   0,  15,  61,  48
1340 DATA   0,  61, 240,   0,  24,   0
1350 DATA   0, 190,   0,   0,  60,   0
1360 DATA   0,  36,   0,   0,  36,   0
1370 DATA   0,  36,   0,   0,  38,   0
1380 DATA   0,  96,   0,  -1
1390 :
1400 DATA   0,  32,   0,   0,   2,   0
1410 DATA   0,   0,   0,   2,   0,   0
1420 DATA   0,   0,  32,   0,  60,   0
1430 DATA   0,  60,   0,   0,  60,   0
1440 DATA   8,  60,   0,   0,  24,  16
1450 DATA   0, 255, 152,   1, 188, 248
1460 DATA  13,  60,   0,  15,  24,  64
1470 DATA   0,  60,   0,   0,  60,   0
1480 DATA   0,  36,   0,   0,  36,   0
1490 DATA   0,  36,   0,   0,  38,   0
1500 DATA   0,  96,   0,  -1
1510 :
1520 :
1530 REM ** FISSA I CONTROLLI DEGLI SPRITE
1540 :
1550 PRINT "]"          :REM PULISCE LO SCHERMO
1560 VIC = 53248        :REM CHIP GRAFICO
1570 :
1580 POKE 2040,13       :REM SPR 0 PUNT DATI
1590 POKE VIC,160       :REM SPR 0 POS ORIZZ
1600 POKE VIC+1,129     :REM SPR 0 POS VERT
1610 POKE VIC+39,1      :REM SPR 0 BIANCO
1620 POKE VIC+23,1      :REM SPR 0 A DOPPIA
1630 POKE VIC+29,1      :REM DIMENSIONE
```

```
1640 POKE VIC+21,1   :REM SPR 0 ATTIVATO
1650 :
1660 :
1670 REM ** ESIBIZIONE
1680 :
1690 IMAGE = PEEK (2040) + 1
1700 IF IMAGE = 16 THEN IMAGE = 13
1710 POKE 2040, IMAGE
1720 :
1730 FOR DELAY = 1 TO 30 : NEXT DELAY
1740 :
1750 :
1760 REM ** ASPETTA LA PRESSIONE DI UN TASTO PER FINIRE
1770 :
1780 GET KP$
1790 IF KP$ = "" THEN 1690
1800 :
1810 POKE VIC+21,0
1820 POKE VIC+29,0
1830 POKE VIC+23,0
1840 :
1850 END
```

FIG. 3.12 Listato del programma "Giocoliere pazzo".

- un occhio che si apre;
- una linea che cresce e si restringe;
- una faccia che sorride;
- una stella che brilla;
- una bufera di neve.

Stabilite i numeri di codice per ogni immagine.

Il giocoliere pazzo

La figura 3.12 è un listato del programma "Giocoliere pazzo", che visualizza le immagini raffigurate nella figura 3.11. Facciamo una breve analisi di alcune sue componenti.
Le istruzioni 1100-1130 caricano i dati di definizione degli sprite. L'istruzione 1120 è un trucco interessante:

```
1120 :   IF SPDTA = -1 THEN PRINT " ."; : GOTO 1140
```

Le definizioni degli sprite riempiono 63 locazioni di memoria, ma sono immagazzinate a intervalli di 64 locazioni di memoria (vedi a pagina).
Se state riempiendo blocchi di memoria adiacenti, potete mantenere semplice il ciclo di caricamento aggiungendo alle liste dei dati un 64-esimo byte con un dato fittizio.
In questo modo, i dati per le immagini degli sprite possono essere caricati consecutivamente. Se scegliete per il byte fittizio un valore che

normalmente non può risultare, potete riconoscerlo e stampare un segnale di caricamento. In questo programma, il valore fittizio è − 1; quando viene letto, il programma aggiungerà un punto alla stampa. Il punto ci dice che un altro blocco è stato letto nella memoria.
I blocchi di dati dell'immagine dei tre sprite sono nelle locazioni 832-894, 896-958, 960-1022.
Dividendo l'indirizzo iniziale di ogni blocco per 64, otterrete i valori del puntatore dello sprite, rispettivamente 13, 14 e 15. Il programma eseguirà la sua animazione cambiando continuamente il valore del puntatore per lo sprite 0, che è fissato inizialmente alla locazione 2040. Il valore andrà da 13 a 14 e a 15, poi ritornerà a 13 per un altro ciclo.
Le istruzioni 1580-1640 fissano i valori iniziali per i controlli dello sprite. Qui non c'è nulla di nuovo. Il puntatore dei dati per lo sprite 0 inizia a 13; l'immagine iniziale sarà quella immagazzinata nelle locazioni di memoria 832-894.
Le istruzioni 1690-1710 commutano le immagini:

```
1690 IMAGE = PEEK (2040) + 1
1700 IF IMAGE = 16 THEN IMAGE = 13
1710 POKE 2040, IMAGE
```

L'istruzione 1690 prende il valore corrente del puntatore e lo incrementa di 1. Se il nuovo valore è 16, la linea 1700 lo riporta a 13. Poi l'istruzione 1710 inserisce il nuovo valore nella locazione del puntatore. Così il puntatore farà quello che noi vogliamo che faccia, cioè andrà da 13 a 14 a 15 e poi ritornerà a 13.
L'istruzione 1730 è un semplice ciclo di ritardo; cambiando la lunghezza della pausa, il giocoliere lancerà la pallina a diverse velocità. E infine, le istruzioni 1780-1790 controllano se è stato premuto un tasto. Se nessun tasto è stato premuto, il programma torna alla linea 1690 per stampare la figura successiva; altrimenti, il programma pulisce i valori dello sprite e termina.

Ora tocca a voi

Prendete la scheda di codifica che avete appena creato.
Usate i codici numerici in sostituzione di dati nelle linee 1160-1500 di "Giocoliere pazzo". Poi fate girare il programma. Divertitevi con il proframma finché otterrete un ciclo di animazione di vostro gradimento. Cambiate la sincronizzazione, i dati e l'ordine con cui le immagini sono mostrate. Con queste esplorazioni avrete modo di imparare molto sull'animazione.

CHE COSA ABBIAMO IMPARATO

Ricapitoliamo quello che abbiamo imparato in questo capitolo:

– Come inizializzare i registri di VIC-II per stampare uno sprite in 4 colori
– Come disegnare uno sprite multicolore
– Che cosa succede quando gli sprite si sovrappongono
– Ulteriori istruzioni per movimenti di più sprite
– Come realizzare un ciclo di animazione spostando un puntatore ai dati dello sprite dal blocco di un'immagine a un altro.

A questo punto, potete solo iniziare a studiare le tecniche per i grafici dello sprite.
Otterrete una conoscenza più approfondita, quando avrete giocato con gli sprite per un po' di tempo. Nei prossimi due capitoli vedremo altri 2 tipi di "magia grafica" del Commodore 64: la grafica a caratteri e a mappa di bit.

ESERCIZI

Test

Le risposte sono elencate più avanti.

1. Nel modo sprite multicolori usare 2 bit significa che un pixel di dimensione doppia assume uno fra .... possibili colori.
2. Dato che gli sprite nel modo multicolore hanno un diametro massimo di 12 pixel di dimensione doppia diciamo che hanno una risoluzione .... inferiore.
3. Se inserite il valore 15 nel registro di selezione dello sprite multicolore in VIC + 28, quali sprite verranno stampati in modo multicolore?
4. Quando i percorsi degli sprite si incrociano, lo sprite .... ha priorità di visualizzazione fra tutti gli altri sprite.
5. Nel programma "Sovrapposizione di sprite" descrivete gli sprite che risultano se inserite queste 3 linee:

```
1100 FOR N = 832 TO 894 STEP 3
1105 :    POKE N, 225
1110 :    POKE N+1, 195
1115 :    POKE N+2, 135
```

6. Guardate le linee 1610-1620 del programma "Sovrapposizione di sprite". Se COUNT ha il valore 120, e DF ha valore 0, che valore assegneranno le linee 1610-1620 e SPRNUM?
7. Quanti punti verranno stampati dopo le parole MI STO PREPARANDO quando i dati dello sprite vengono inseriti durante il programma "Giocoliere pazzo"?
8. Cosa succede al giocoliere nel programma suddetto se si cambia il tempo di ritardo nell'istruzione 1730 da 30 a 100?

Esercizi di programmazione

1. Cambiate il programma "Sprite in 4 colori" in modo che un secondo sprite basato sugli stessi dati sia visualizzato in modo multicolore.
2. Cambiate il programma "Sovrapposizione di sprite" in modo che i quattro sprite si sovrappongano completamente al centro dello schermo.

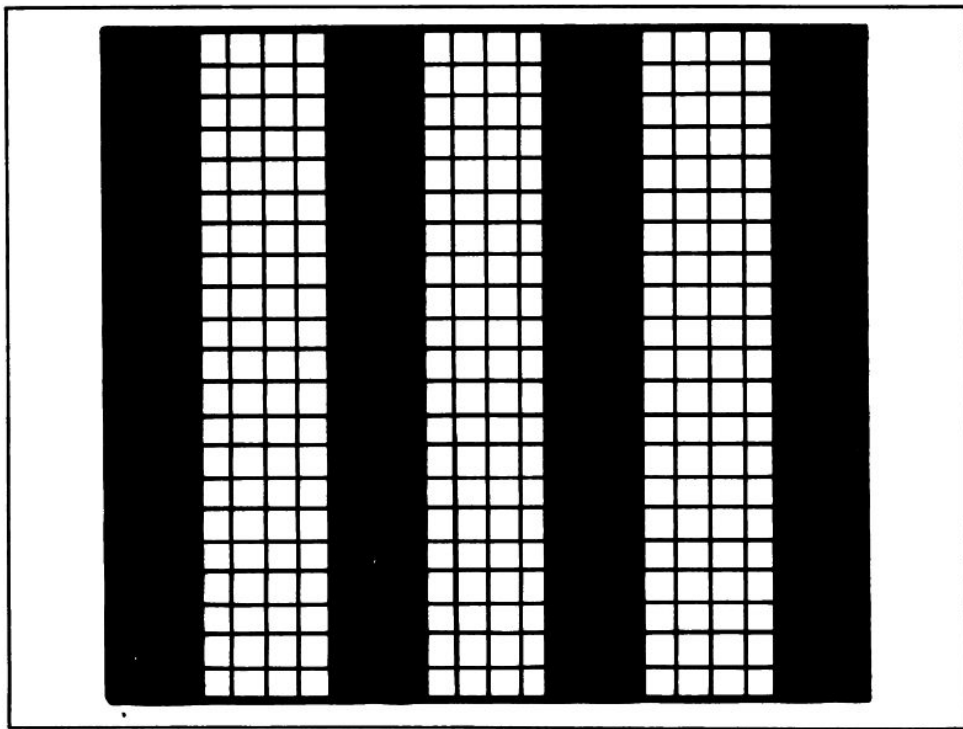



FIG. 3.13. Sprite che risultano dopo aver inserito i cambiamenti nel programma "Sovrapposizione" citati nel test 5.

3. Cambiate il programma "Giocoliere pazzo" in modo che lanci le palline in direzione oraria per un certo periodo di tempo, poi in direzione antioraria e poi ancora oraria e così via.

Soluzioni del test

1. 4
2. orizzontale
3. 0, 1, 2, 3
4. 0
5. ogni sprite è formato da 4 strisce verticali (vedi figura 3.13)
6. 2
7. 3
8. Il gioco rallenta

Soluzioni possibili agli esercizi di programmazione

Queste soluzioni si basano sull'aggiunta o il cambiamento di alcune istruzioni nei programmi originali.

1. Caricate il programma "Sprite 4 colori". Poi inserite queste istruzioni:

```
1000 REM *** DUE A 4 COLORI ***
1315 POKE 2041,14    :REM PUNTATORE SPRITE 1
1353 POKE VIC+2,160  :REM POS ORIZZONTALE SPRITE 1
1356 POKE VIC+3,69   :REM POS VERTICALE SPRITE 1
1370 POKE VIC+23,3   :REM ESPANSIONE VERTICALE
1380 POKE VIC+29,3   :REM ESPANSIONE ORIZZONTALE
1400 POKE VIC+28,3   :REM MULTICOLORE 0 E 1
1435 POKE VIC+40,2   :REM SPRITE 1 ROSSO
1460 POKE VIC+21,3   :REM SPRITE 0 E 1 ATTIVATI
```

2. Caricate il programma "Sovrapposizione". Poi inserite queste istruzioni:

```
1000 REM *** SOVRAPPOSIZIONE TOTALE ***
1250 POKE VIC,210    :REM SPR 0 POS OR
1260 POKE VIC+2,110  :REM SPR 1 POS OR
1270 POKE VIC+4,160  :REM SPR 2 POS OR
1280 POKE VIC+6,160  :REM SPR 3 POS OR
1300 POKE VIC+1,129  :REM SPR 0 POS VERT
1310 POKE VIC+3,129  :REM SPR 1 POS VERT
1320 POKE VIC+5,179  :REM SPR 2 POS VERT
1330 POKE VIC+7,79   :REM SPR 3 POS VERT
```

3. Caricate il programma "Giocoliere pazzo". Poi inserite queste istruzioni:

```
1000 REM *** GIOCOLIERE ALTERNATO ***
1655 GIOCDIR = 1     :REM SENSO ORARIO
1690 IMAGE = PEEK (2040) + GIOCDIR
1705 IF IMAGE = 12 THEN IMAGE = 15
1712 :
1715 COUNT = COUNT + 1
1718 IF INT (COUNT/27) = COUNT/27 THEN GIOCDIR = - GIOCDIR:COUNT = 0
```

CAPITOLO QUATTRO

Grafica a caratteri

Il Commodore 64 ha ottime capacità di visualizzazione di testi. In questo capitolo ne vedremo alcune. Tratteremo gli insiemi di caratteri già esistenti e assegneremo valori nella memoria dello schermo e nella memoria dei colori.
Creeremo stringhe di caratteri grafici. Impareremo a modificare l'insieme dei caratteri già esistenti, e infine vedremo come disegnare un insieme di caratteri da usare per l'animazione.

GIOCHIAMO UN PO'

È ora di fare una piccola esplorazione della tastiera. Sedetevi al vostro Commodore 64. Inserite questo comando:

```
POKE 650, 128
```

Nel caso non lo sappiate, l'inserimento di un numero maggiore di 127 nella locazione di memoria 650 causa la ripetizione dei tasti digitati, quando questi vengono tenuti premuti per un tempo abbastanza lungo.
È divertente disegnare con i tasti che vengono ripetuti. Per ritornare alla situazione normale, dove si ripetono solo alcuni tasti, inserite uno 0 nella stessa locazione.
Ora pulite il video. Pensate che il video della vostra TV sia una tela bianca per dipingere. Usando i vari caratteri grafici, inserite alcuni disegni.

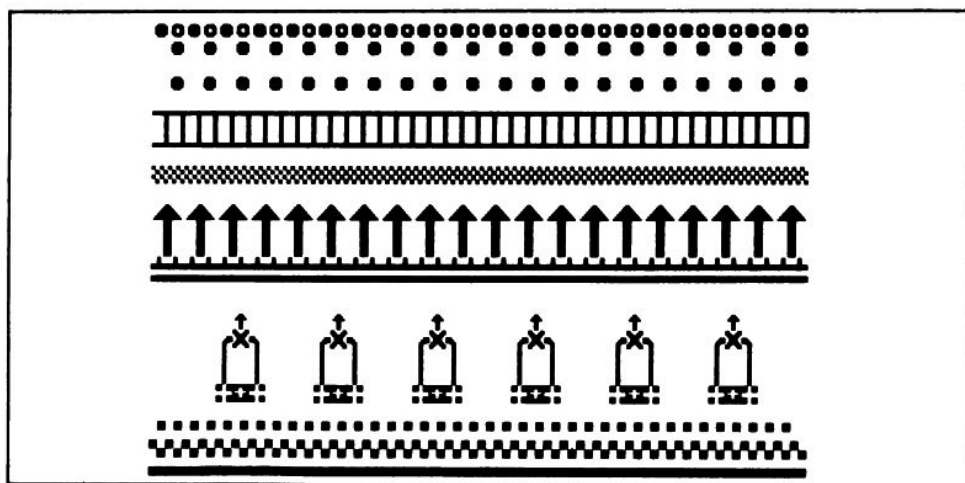



FIG. 4.1. Visualizzazione di un disegno sullo schermo con alcuni dei 512 caratteri già esistenti nel Commodore 64.

Alcuni tasti vi saranno particolarmente utili: quello delle maiuscole (SHIFT), il tasto con il simbolo Commodore, CTRL (controllo), i tasti del colore, RVS (reverse) ON e RVS OFF e i tasti di controllo cursore. Ci sono 512 caratteri diversi già inseriti nella memoria permanente del Commodore 64: potete quindi ottenere con questa semplice tecnica alcuni disegni interessanti. La figura 4.1 mostra uno di questi disegni.

LA MEMORIA DEL VIDEO E DEL COLORE

Il Commodore 64 visualizza normalmente 25 linee di testo, ognuna delle quali contiene 40 caratteri.
Questo ci dà 1000 locazioni di schermo. Le codifiche che determinano quale carattere è mostrato in ciascuna locazione sono immagazzinate nella cosiddetta «memoria del video». Il Commodore 64, con la sua flessibilità, consente di spostare questa memoria dello schermo, la quale normalmente occupa le 1000 locazioni di memoria da 1024 a 2023.
Esiste un secondo blocco di 1000 locazioni di memoria che controlla il colore per ogni locazione del video.
Questa area di memoria, chiamata "memoria del colore", occupa le locazioni di memoria 55296-56295. La memoria del colore è un po' scarsa; ogni locazione può contenere solo 4 bit, il che limita agli interi da 0 a 15. Ma ci sono solo 16 colori possibili, quindi è sufficiente.
Ogni locazione sul video per i testi ha associate normalmente 2 locazioni

in memoria. Una, nella memoria del video, determina quale dei 256 caratteri sarà visualizzato; l'altra, nella memoria del colore, determina il colore che verrà attribuito al carattere. Le appendici B e C illustrano le aree di memoria del video e del colore.

CARATTERI SULLO SCHERMO

L'integrato VIC-II controlla la presentazione dei caratteri sul video. Esso esplora ogni secondo la memoria del video. Queste locazioni contengono valori fra 0 e 255. Basandosi sui valori trovati in quelle locazioni, VIC-II entra nella sezione di memoria dove sono immagazzinati i modelli per disegnare tutti i diversi caratteri. Utilizza quei modelli e l'informazione nelle locazioni di memoria del colore per mandare i segnali elettrici corretti alla TV.
Il Commodore 64 ha modelli per due insiemi completi di caratteri immagazzinati in una parte della sua memoria permanente. Ogni insieme contiene il modello per 256 caratteri. Il dispositivo in cui sono immagazzinati gli insiemi è il generatore di caratteri in ROM. Diamo uno sguardo a questi caratteri già inseriti.

VISUALIZZAZIONE DEI 512 CARATTERI INTERNI

La figura 4.2 è un listato del programma "Caratteri in ROM". Inseritelo nel computer, salvatelo e poi fatelo girare. Quando inizia la visualizzazione compaiono i primi 256 caratteri. Per vedere gli altri 256, dovete digitare nello stesso momento i tasti **SHIFT Commodore**. Agiscono da commutatore tra i due insiemi di caratteri.
Il funzionamento del programma è semplice in linea di principio, ma un po' più complesso nell'esecuzione. Vogliamo soltanto inserire ognuno dei valori tra 0 e 255 nella locazione della memoria del video. Questo è ciò che esegue il ciclo delle linee **1140-1260**.
Le difficoltà iniziano quando si calcolano le locazioni per ottenere una stampa piacevole. Questo è quello che fanno le istruzioni **1150-1220**.
Le linee **1240-1250** fanno il lavoro di inserimento:

```
1240 :   POKE SCRMAP + SPOT, POCODE
1250 :   POKE COLMAP + SPOT, 1
```

```
1000 REM *** CARATTERI IN ROM ***
1010 :
1020 :
1030 REM ** PULISCE LO SCHERMO E
1040 REM    INIZIALIZZA LE COSTANTI
1050 :
1060 PRINT "□"
1070 :
1080 SCRMAP = 1024
1090 COLMAP = 55296
1100 :
1110 :
1120 REM ** CICLO DI VISUALIZZAZIONE
1130 :
1140 FOR POCODE = 0 TO 255
1150 :   ROW = INT (POCODE / 20)
1160 :   CLM = POCODE - (20 * ROW)
1170 :
1180 :   EVROW = (ROW/2 = INT(ROW/2))
1190 :   CLM = (CLM * 2) - EVROW
1200 :   ROW = ROW * 2
1210 :
1220 :   SPOT = (ROW * 40) + CLM
1230 :
1240 :   POKE SCRMAP + SPOT, POCODE
1250 :   POKE COLMAP + SPOT, 1
1260 NEXT POCODE
1270 :
1280 :
1290 REM ** ASPETTA UN TASTO PER FINIRE
1300 :
1310 GET KP$
1320 IF KP$ = "" THEN 1310
1330 :
1340 PRINT "□";
1350 END
```

FIG. 4.2. Listato del programma "Caratteri in ROM".

Oltre a mettere il codice di un carattere nella memoria dello schermo, si inserisce il valore 1 nella corrispondente locazione della memoria del colore. In questo modo, il carattere sarà visualizzato con il colore 1, bianco.
Potreste creare qualche variazione: visualizzare caratteri in colori diversi oppure farli visualizzare in locazioni diverse.

COSTRUIRE UNA STRINGA DI CARATTERI E SPOSTARLA

La figura 4.3 mostra il listato del programma "Faccia che vola".
Il programma ci fa vedere come si costruiscono, con i caratteri, figure che si muovono. Inserite il programma nel computer e fatelo girare.
Digitando uno dei tasti di controllo del cursore (su, giù, destra, sinistra),

```
1000 REM *** FACCIA CHE VOLA ***
1010 :
1020 :
1030 REM ** COSTRUISCE LA STRINGA
1040 :
1050 F$(1) = "          "
1060 F$(2) = " ┌──────┐ "
1070 F$(3) = " | O O | "
1080 F$(4) = " |  ●  | "
1090 F$(5) = " | ⌣ | "
1100 F$(6) = " └──────┘ "
1110 F$(7) = "          "
1120 :
1130 D1L9$ = "█████████████"
1140 U7$ = "TTTTTTT"
1150 :
1160 FOR N = 1 TO 7
1170 :    F$ = F$ + F$(N) + D1L9$
1180 NEXT N
1190 F$ = F$ + U7$
1200 :
1210 :
1220 REM ** INIZIA A CENTRO SCHERMO
1230 :
1240 PRINT "█████████████████████████████";
1250 PRINT F$;   : REM STAMPA LA FACCIA
1260 :
1270 :
1280 REM ** ASPETTA UN TASTO PREMUTO
1290 :
1300 POKE 650,128   :REM TUTTI I TASTI RIPETONO
1310 GET KP$
1320 IF KP$ = "" THEN 1310
1330 :
1340 :
1350 REM ** DECIFRA IL TASTO PREMUTO
1360 :
1370 IF KP$ = "█" THEN 1440   :REM SU
1380 IF KP$ = "█" THEN 1440   :REM GIU'
1390 IF KP$ = "█" THEN 1440   :REM DESTRA
1400 IF KP$ = "█" THEN 1440   :REM SINISTRA
1410 IF KP$ = " " THEN 1500   :REM SPAZIO
1420 GOTO 1310    :REM NON CORRISPONDE
1430 :
1440 PRINT KP$ ;  :REM SPOSTA IL CURSORE
1450 GOTO 1250    :REM STAMPA LA FACCIA
1460 :
1470 :
1480 REM ** PER FINIRE, LA BARRA SPAZIO
1490 :
1500 PRINT "█";    :REM PULISCE TUTTO
1510 END
```

FIG. 4.3. Listato del programma "Faccia che vola".

la faccia sorridente si muoverà; il programma si chiude premendo la barra spazio.

Creazione della stringa

La prima parte del prógramma costruisce una stringa speciale. Questa stringa, chiamata F$, contiene spazi bianchi, caratteri grafici e comandi di spostamento del cursore. Quando questa stringa è stampata, la faccia sorridente sarà visualizzata sul video come mostrato nel listato.
Le istruzioni 1050-1140 precisano i pezzi che saranno inseriti in F$. Le istruzioni 1160-1190 li uniscono:

```
1160 FOR N = 1 TO 7
1170 :    F$ = F$ + F$(N) + D1L9$
1180 NEXT N
1190 F$ = F$ + U7$
```

Ogni pezzo di grafica è seguito da un pezzo di movimento del cursore. Vengono stampati alcuni caratteri, poi il cursore scende di una riga e torna a sinistra. L'istruzione 1190 aggiunge un movimento finale del cursore per riportare il cursore alla sua posizione iniziale.

Come si muove la stringa

L'istruzione 1240 pulisce il video, e poi posiziona il cursore a metà schermo. L'istruzione 1250 disegna la stringa della faccia che abbiamo costruito:

```
1250 PRINT F$;   : REM STAMPA LA FACCIA
```

Infine, il programma entra nella fase di movimento.
L'istruzione 1300 inizializza la tastiera per l'autoripetizione. Le istruzioni 1310-1320 aspettano che voi premiate un tasto. Quando ne premete uno, viene memorizzato in KP$.
Le linee 1370-1420 decifrano KP$:

```
1370 IF KP$ = "▔" THEN 1440  :REM SU
1380 IF KP$ = "▒" THEN 1440  :REM GIU'
1390 IF KP$ = "▒" THEN 1440  :REM DESTRA
1400 IF KP$ = "▒" THEN 1440  :REM SINISTRA
1410 IF KP$ = " " THEN 1500  :REM SPAZIO
1420 GOTO 1310   :REM NON CORRISPONDE
```

Se il tasto è uno dei 4 movimenti del cursore (su, giù, sinistra o destra), il programma salta alla linea 1440. Se digitate lo spazio il programma salta alla linea 1500 per poi terminare. Se il tasto premuto non è fra questi, il programma torna solamente a leggere la linea 1310.
Che cosa succede se digitate uno dei tasti di controllo del cursore?

```
1440 PRINT KP$ ;  :REM SPOSTA IL CURSORE
1450 GOTO 1250    :REM STAMPA LA FACCIA
```

Stamperete ciò che corrisponde al tasto digitato, che muove il cursore. Poi il programma ritorna alla linea 1250, stampa la faccia nella sua nuova posizione e continua a girare aspettando che venga premuto un altro tasto.

Trasferimento del cancellino

Vi sarete chiesti il motivo per cui la faccia sia stata disegnata circondata da un anello di spazi. Questa è la tecnica del "trasporto del cancellino personale". La faccia può muoversi solo di una posizione alla volta. Non vi preoccupate di cancellare la vecchia faccia quando la muovete in una nuova posizione. Quando la faccia si trova in una nuova posizione, copre la maggior parte della vecchia faccia e l'anello di spazi esterno copre qualsiasi parte restante.
Se volete muovere la faccia di 2 posizioni, l'anello di spazi dovrà essere largo 2 spazi.
Se non avessimo usato questa tecnica, avremmo dovuto cancellare completamente la faccia dalla sua vecchia posizione prima di disegnare la nuova faccia.
Quest'ultimo metodo fa perdere molto tempo; nell'animazione, invece, si cerca sempre di muovere e disegnare gli oggetti nel modo più veloce possibile.

Movimento della faccia

È ora di applicare ciò che abbiamo imparato giocando con la tastiera agli inizi del capitolo. Cambiate le linee 1050-1190, così si muoverà sullo schermo una faccia diversa. Se la volete più elegante, inserite alcuni caratteri di controllo del colore nella vostra stringa. Cercate di aggiungere altre funzioni attivate dai tasti. Tanto per divertirvi, create una faccia che non sia circondata dall'anello di spazi di autocancellazione.

ANCORA SULLA MEMORIA DEI CARATTERI

Quando accendete il vostro Commodore 64, esso va a prendere i modelli dei caratteri nel generatore interno, in ROM. Una ROM è una

memoria di sola lettura. I modelli del carattere vengono inseriti nella ROM quando essa viene prodotta. Non si possono inserire nuove informazioni in ROM.
Comunque potete dire all'integrato VIC-II che prenda i suoi modelli da altre aree di memoria. Queste aree possono essere la memoria RAM, in cui, oltre alla lettura, è permessa la scrittura. Così potete inserire i vostri modelli di caratteri usando il VIC-II.
Il VIC-II controlla 16K (16.384 byte) di memoria alla volta. Un insieme completo di modelli per 256 caratteri occupa 2K, 2048 byte di memoria. Ci sono quindi 8 possibili locazioni per il blocco di memoria dei caratteri da 2K, in un banco di 16K.
I bit 1, 2, 3 del registro localizzato in VIC + 24 (53272) dicono al VIC-II dove trovare i modelli del carattere. Quando si accende la macchina, esso guarda al blocco da 2K che inizia alla locazione 4096 e trova i primi 256 modelli immagazzinati nel generatore di caratteri. Se premete il tasto SHIFT insieme al tasto Commodore, vengono immagazzinati nuovi valori in VIC + 24. VIC-II ora guarda il blocco di 2K dei modelli di caratteri che inizia nella locazione 6144 e stampa i caratteri dal secondo insieme di 256 caratteri immagazzinato nella ROM.
Se volete usare altri caratteri dovete riempire un blocco di 2K di RAM coi modelli e poi inizializzare i puntatori nei bit 1, 2, 3 di VIC + 24. Il modello per ogni carattere usa 8 byte; è un grosso lavoro rappresentare i modelli per un insieme completo di 256 caratteri. Esiste comunque una scorciatoia.
In molti casi accade che si vogliano cambiare solo alcuni modelli di carattere; allora si può copiare nella memoria RAM un insieme di modelli del generatore di caratteri in ROM, per poi cambiarne soltanto alcuni.

CARATTERI DALLA ROM ALLA RAM

Esistono alcune complicazioni per lo spostamento da ROM in RAM. In primo luogo, la ROM dei caratteri è un po' sfuggente: perde un sacco di tempo a comparire in differenti locazioni di memoria. Ora è in un posto, ora in un altro. Dovete chiuderla in un'area precisa abbastanza a lungo da poter copiare i suoi contenuti.
Questo ci porta alla seconda complicazione: quando cercate di "bloccare" la ROM, si inserisce nell'area di memoria che normalmente usano i dispositivi di input/output del Commodore.
Con la ROM in memoria il computer non può comunicare con il mondo esterno. Se tenta di eseguire operazioni di I/O non andrà da nessuna parte.

```
1000 REM *** ROM IN RAM ***
1010 :
1020 :
1030 REM ** PROVVEDE UN FEEDBACK
1040 :
1050 PRINT "[illegible]STO TRASFERENDO",
1060 :
1070 :
1080 REM ** PREPARA PER IL TRASFERIMENTO
1090 :
1100 POKE 56334, PEEK (56334) AND 254
1110 REM ** DISATTIVA INTERRUPT SCANSIONE TASTIERA
1120 :
1130 POKE 1, PEEK (1) AND 251
1140 REM ** PORTA LA ROM IN MEMORIA
1150 :
1160 ROM = 53248 :REM INIZIO ROM CARATTERI
1170 RAM = 12288 :REM DESTINAZIONE
1180 :
1190 :
1200 REM ** TRASFERIMENTO, CON FEEDBACK
1210 :
1220 FOR CHAR = 0 TO 255
1230 :    SR = ROM + (CHAR * 8)
1240 :    DS = RAM + (CHAR * 8)
1250 :
1260 :    FOR BYTE = 0 TO 7
1270 :    POKE DS + BYTE, PEEK (SR + BYTE)
1280 :    NEXT BYTE
1290 :
1300 :    POKE 1, PEEK(1) OR 4
1310 :    PRINT ".";
1320 :    POKE 1, PEEK(1) AND 251
1330 NEXT CHAR
1340 :
1350 :
1360 REM ** PULISCE TUTTO
1370 :
1380 POKE 1, PEEK (1) OR 4
1390 POKE 56334, PEEK (56334) OR 1
1400 :
1410 VIC = 53248 : CPTR = VIC+24
1420 PTR = PEEK (CPTR) AND 241
1430 PTR = PTR OR 12
1440 POKE CPTR, PTR
1450 :
1460 PRINT : PRINT "FATTO."
1470 END
```

FIG. 4.4. Listato del programma "ROM in RAM".

Ora c'è un'operazione che il vostro Commodore cerca di fare 60 volte al secondo: scandire la tastiera. Dovete sospendere quell'operazione quando trasferite la ROM in RAM. È come bloccare le arterie durante un'operazione.

UN ESEMPIO PRATICO

La figura 4.4 mostra il listato del programma "ROM in RAM". Vediamo come viene effettuato il trasferimento.
La linea 1100 disattiva la scansione della tastiera:

```
1100-POKE 56334, PEEK (56334) AND 254
```

Questa istruzione inserisce uno 0 nel bit 0 della locazione 56334, e lascia stare gli altri bit. Essa blocca l'operazione di scansione della tastiera. Consultate l'appendice N per ulteriori informazioni sull'istruzione AND. La linea 1130 ferma la ROM nella memoria, perché sia possibile copiarla:

```
1130 POKE 1, PEEK (1) AND 251
```

Questa istruzione inserisce uno 0 nel bit 2 della locazione 1, lasciando ancora intatti gli altri bit. Quel bit causa l'inserimento della ROM nella memoria. Durante questo processo, le funzioni di I/O della macchina sono messe da parte. Ancora, i lettori più curiosi possono consultare l'appendice N per dettagli sul funzionamento dell'istruzione AND.
Le linee 1220-1330 trasferiscono i primi 256 modelli di caratteri (2048 byte) da ROM in RAM. Ci vuole un po' di tempo, e il programma dà qualche indicazione dei lavori in corso.
Il blocco viene trasferito in 256 pezzi, 8 byte alla volta. La linea 1270 determina il trasferimento effettivo:

```
1270 :        POKE DS + BYTE, PEEK (SR + BYTE)
```

Essa prende (PEEK) il valore in una locazione di memoria ROM e poi inserisce il valore che ha trovato in una locazione di memoria RAM.
Dopo il trasferimento di ogni gruppo di 8 byte, il programma stampa un punto (.) sul video. Per questo è necessario ritornare un attimo alla funzione I/O:

```
1300 :    POKE 1, PEEK(1) OR 4
1310 :    PRINT ".";
1320 :    POKE 1, PEEK(1) AND 251
```

La linea 1300 inserisce un 1 nel bit 2 della locazione 1 della memoria, cioè ristabilisce le funzioni di I/O. L'appendice N ci parla anche dell'i-

struzione OR. La linea 1310 stampa il punto. La linea 1320 poi esclude ancora l'I/O e riinserisce la ROM.
Quando tutti i 2048 byte sono stati copiati nella memoria RAM, la linea 1380 ristabilisce definitivamente l'I/O.
La linea 1390 fa ripartire la scansione della tastiera inserendo un 1 nel bit 0 della locazione 56334. Infine le linee 1410-1440 dicono a VIC-II di iniziare a usare le nuove locazioni di memoria RAM per i modelli di caratteri:

```
1410 VIC = 53248 : CPTR = VIC+24
1420 PTR = PEEK (CPTR) AND 241
1430 PTR = PTR OR 12
1440 POKE CPTR, PTR
```

Queste linee potrebbero sembrare un po' enigmatiche.
Vediamo come funzionano.
Tre bit del registro di VIC + 24 controllano la locazione dei modelli dei caratteri: i bit 1, 2, 3. Il bit 0 di quel registro non fa nulla. Quando volete cambiare le locazioni dei modelli dei caratteri, prima pulite i bit 1, 2, 3 e poi attribuite loro nuovi valori.
La linea 1420 pulisce i 3 bit in questione tramite un'operazione di AND, che attribuisce il valore 0 ai bit 1, 2, 3 senza toccare gli altri. Poi la linea 1430 dà ai bit un nuovo valore tramite un'operazione di OR.
I blocchi di memoria contenenti i modelli dei caratteri devono iniziare nelle locazioni di memoria che sono multipli di 2048. In questo caso, i modelli iniziano in 12288, cioè 6 × 2048.
Quando volete far puntare VIC a un blocco di modelli dei caratteri, dividete l'indirizzo iniziale per 1024 e poi usate il risultato per attribuire i valori ai bit in VIC + 24. 12288 : 1024 = 12, quindi questo è il numero da usare per dare i valori ai bit.

UNA PICCOLA MODIFICA

Se non l'avete ancora fatto, inserite e fate girare il programma "Trasferimento del carattere da ROM in RAM". Nulla sembra accadere quando il programma termina.
Digitate i tasti SHIFT Commodore per raggiungere il secondo insieme di caratteri: sorpresa!
Avete trasferito in RAM solo un insieme di modelli dei caratteri. Quando commutate gli insiemi, VIC cerca i modelli nel successivo blocco di 2K di RAM. Dato che non avete inserito modelli in quel blocco, le lettere

diventano degli sgorbi irregolari. Premete il tasto SHIFT e il tasto Commodore per ritornare al primo insieme.
Facciamo alcuni cambiamenti dei modelli. Inserite questi comandi, uno alla volta, e guardate come cambia la parola READY sul vostro schermo:

```
POKE 12296, 238
POKE 12297, 204
POKE 12298, 204
POKE 12299, 252
POKE 12300, 204
POKE 12301, 216
POKE 12302, 112
POKE 12303, 0
```

Avete cambiato il modello usato dal VIC per visualizzare la lettera A sul video. Ogniqualvolta la codifica di A appare nella memoria del video, VIC userà il nuovo modello per disegnare la lettera.
Questo comando dirà al VIC di usare ancora i modelli della ROM originaria:

```
POKE 53272, 21
```

Inserite il comando, e guardate la vostra A che torna normale.
Per ottenere ancora i caratteri alternativi, usate questo comando scorciatoia che dice al VIC di usare i modelli che avete messo in RAM a partire nella locazione 12288:

```
POKE 53272, 29
```

DISEGNARE I CARATTERI

La figura 4.5 mostra una scheda di codifica che potete usare per disegnare un carattere. È molto simile alle schede di codifica che abbiamo usato per gli sprite. Sono usati otto byte per codificare un carattere. Ogni byte codifica la configurazione dei pixel per una riga del carattere. Ogni bit in un byte rappresenta un pixel.
In ogni riga, i valori dei bit dei pixel che saranno visualizzati vengono sommati per ottenere un numero di codice.
Le figure 4.6 e 4.7 sono esempi che mostrano l'utilizzazione di queste schede di codifica. Nella figura 4.6 si vede la codifica di una A normale.

Numero del bit	7	6	5	4	3	2	1	0	Numero codici
Valore del bit	128	64	32	16	8	4	2	1	
Byte 0									
Byte 1									
Byte 2									
Byte 3									
Byte 4									
Byte 5									
Byte 6									
Byte 7									

FIG. 4.5. Una scheda di codifica che potete usare per disegnare i caratteri.

La figura 4.7 dà, invece, la codifica per una A capovolta più elaborata. Questi codici sono i numeri che avete inserito nel paragrafo precedente. Fate alcune copie della scheda in figura 4.5. Poi disegnate una versione rovesciata della lettera E. La useremo nel prossimo paragrafo.

INSERIMENTO DI MODIFICHE

L'appendice D è un elenco di codici per la visualizzazione sullo schermo.
Questi sono i numeri che vengono inseriti nella memoria del video per dire al VIC-II quale modello del carattere considerare. Per esempio, il codice di visualizzazione per @ è 0 e per A è 1.
Ogni modello di carattere usa 8 byte. I modelli sono immagazzinati in

Numero del bit	7	6	5	4	3	2	1	0	Numero codici
Valore del bit	128	64	32	16	8	4	2	1	
Byte 0									24
Byte 1									60
Byte 2									102
Byte 3									126
Byte 4									102
Byte 5									102
Byte 6									102
Byte 7									0

FIG. 4.6. Esempio di scheda di codifica del carattere compilata.

ordine di codice di visualizzazione. Prima si trovano gli 8 byte per @, poi gli 8 byte per A, e così via. Per trovare la locazione di memoria del primo byte di un modello di 8 byte di carattere, moltiplicate soltanto il codice di visualizzazione del carattere per 8 e sommate il risultato all'inizio del blocco della memoria del carattere.
Facciamo un esempio. Nel programma "ROM in RAM" avevamo trasferito la memoria del carattere a un blocco di 2K a partire da 12288.
Per trovare il primo byte del modello per la lettera A, abbiamo moltiplicato il suo codice di visualizzazione per 8 e abbiamo sommato il risultato a 12288. 1 × 8 = 8 e 12288 + 8 = 12296.
In questo modo le locazioni 12296-12303 (8 byte) contengono i modelli per A. Se riguardate a pagina xx, vedete che quelle sono le 8 locazioni che avevamo inserito per cambiare la forma della A.
Proviamo di nuovo. Il codice di stampa per E è 5. 5 × 8 = 40 e 12288 + 40 = 12328.

Numero del bit	7	6	5	4	3	2	1	0	Numero codici
Valore del bit	128	64	32	16	8	4	2	1	
Byte 0									238
Byte 1									204
Byte 2									204
Byte 3									252
Byte 4									204
Byte 5									216
Byte 6									112
Byte 7									0

FIG. 4.7. Un altro esempio di scheda di codifica compilata.

Fate girare il programma "Trasferimento del carattere da ROM in RAM" e poi inserite nelle 8 locazioni di memoria a partire da **12328** il codice della E capovolta, che abbiamo elaborato nell'ultimo paragrafo. Osservate il **READY** sullo schermo, durante l'operazione.

DISEGNARE UN INSIEME DI CARATTERI PER L'ANIMAZIONE

Avete visto come cambiare i caratteri di testo. Questo vi dà l'abilità di sviluppare ogni tipo di simboli per i giochi elettronici, le applicazioni commerciali, le lingue straniere e le vignette. Guardiamo ora come si possono sviluppare alcuni caratteri per ottenere dei movimenti veloci. Perché usare i caratteri per l'animazione quando gli sprite sono così facili da usare? Ci sono situazioni in cui l'animazione dei caratteri

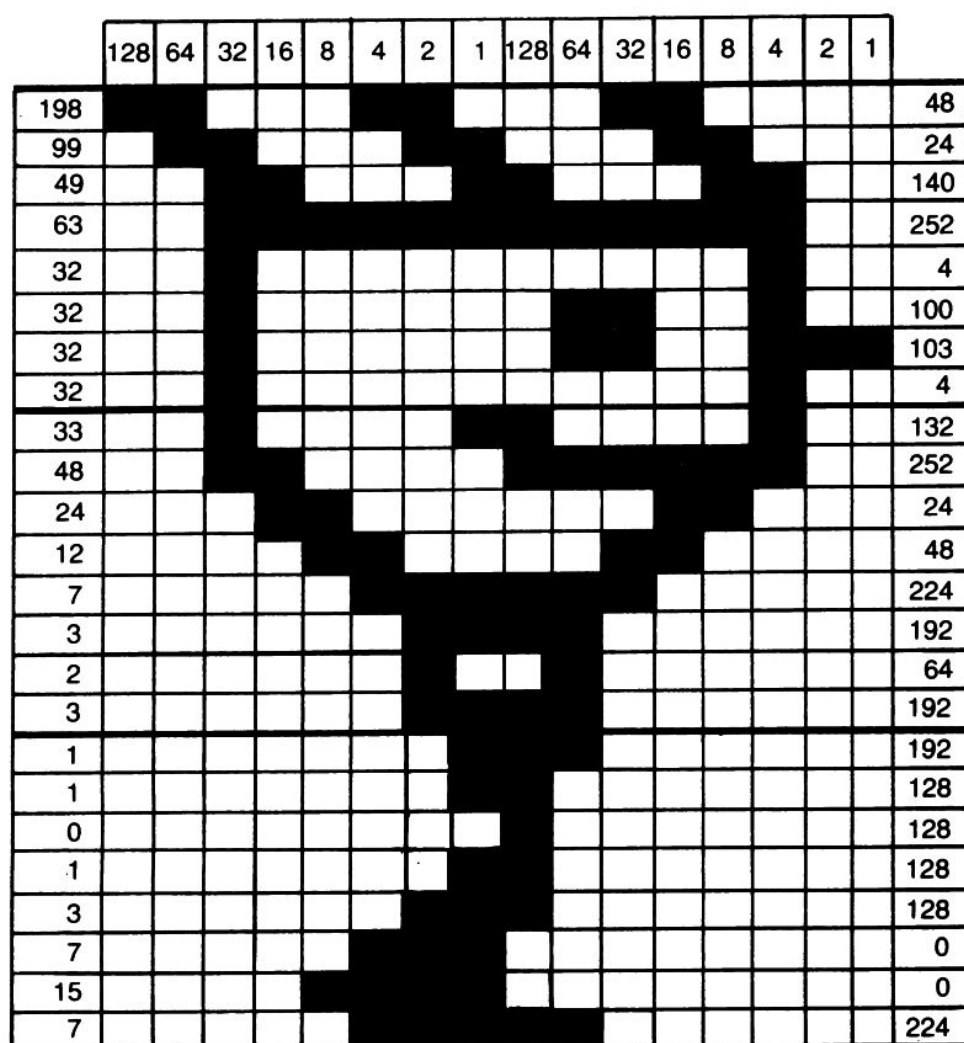



FIG. 4.8. Una creatura aliena disegnata su una griglia alta 3 caratteri e larga 2.

tradizionali presenta qualche utilità. In certi casi gli 8 sprite possono essere già impegnati.
Inoltre l'animazione di un carattere permette diverse variazioni di colore senza perdere la risoluzione orizzontale.
Infine però si è più liberi per quanto riguarda la forma e la dimensione del carattere poiché, si può mettere qualsiasi combinazione di caratteri in un'immagine.

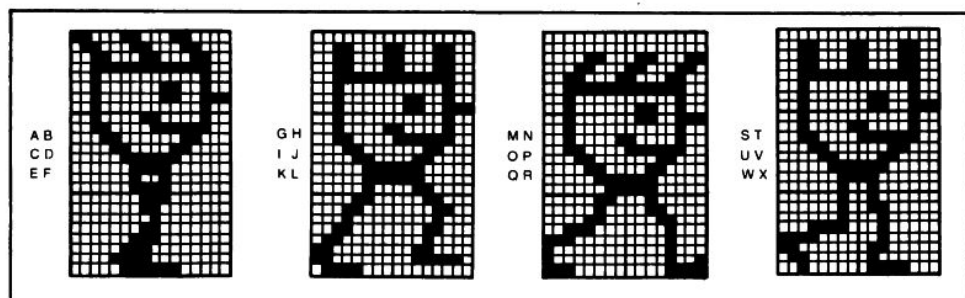



FIG. 4.9. Il disegno dell'alieno in 4 posizioni su griglie 2 × 3, con lettere, elencate a fianco di ogni immagine, per la sostituzione.

La figura 4.8 mostra una creatura aliena, disegnata su una griglia che è larga 2 caratteri e alta 3. In alto, sono dati i valori per ogni posizione dei bit. Di lato è dato il codice per le righe del byte. Per esempio, i codici per il carattere usato nell'angolo destro in basso del disegno sono 192, 128, 128, 128, 128, 0, 0, 224.

La figura 4.9 presenta il nostro alieno in quattro posizioni. Ogni posizione è disegnata su una griglia larga 2 caratteri e alta 3. A fianco di ogni immagine c'è una indicazione sulla tecnica che verrà usata per visualizzare l'alieno sullo schermo.

Inserite la codifica sviluppata dalle immagini al posto delle lettere A-X. Poi stamperete la stringa fatta da quelle lettere combinate con alcuni movimenti del cursore, come nel programma "Spostare la faccia". Anziché stampare blocchi di dimensioni 2 × 3 di lettere vere, il VIC-II mostrerà blocchi di 2 × 3 che rappresentano il nostro alieno che cammina.

L'ALIENO CHE CAMMINA

Il programma "Alieno cammina" è mostrato in figura 4.10. Guardiamo come è strutturato. Abbiamo quattro immagini, ognuna composta da 6 modelli di caratteri ridefiniti, con 8 byte per modello, il che ci dà 24 × 8, cioè 192, byte di dati da caricare.

Le istruzioni 1100-1180 ci caricano:

```
1100 BASE = 12 * 1024
1110 FOR CHAR = 1 TO 24
1120 :    FOR BYTE = 0 TO 7
1130 :         READ INFO
1140 :         SPOT = BASE + (CHAR * 8) + BYTE
1150 :         POKE SPOT, INFO
1160 :    NEXT BYTE
1170 :    PRINT ".";      :REM FEEDBACK
1180 NEXT CHAR
```

```
1000 REM *** ALIENO CAMMINA ***
1010 :
1020 :
1030 REM ** PROVVEDE FEEDBACK SULLO SCHERMO
1040 :
1050 PRINT "[illegible]STO CARICANDO";
1060 :
1070 :
1080 REM ** LEGGE I DATI
1090 :
1100 BASE = 12 * 1024
1110 FOR CHAR = 1 TO 24
1120 :    FOR BYTE = 0 TO 7
1130 :        READ INFO
1140 :        SPOT = BASE + (CHAR * 8) + BYTE
1150 :        POKE SPOT, INFO
1160 :    NEXT BYTE
1170 :    PRINT ".";      :REM FEEDBACK
1180 NEXT CHAR
1190 :
1200 DATA 198,99,49,63,32,32,32,32
1210 DATA 48,24,140,252,4,100,103,4
1220 DATA 33,48,24,12,7,3,2,3
1230 DATA 132,252,24,48,224,192,64,192
1240 DATA 1,1,0,1,3,7,15,7
1250 DATA 192,128,128,128,128,0,0,224
1260 :
1270 DATA 0,49,49,49,63,32,32,32
1280 DATA 0,140,140,140,252,4,100,103
1290 DATA 32,33,48,24,12,7,7,4
1300 DATA 4,132,252,24,48,224,224,48
1310 DATA 12,24,16,48,96,192,192,120
1320 DATA 24,12,8,24,16,48,62,0
1330 :
1340 DATA 0,0,12,24,49,63,32,32
1350 DATA 0,0,99,198,140,252,4,100
1360 DATA 32,32,33,48,24,12,7,3
1370 DATA 103,4,132,252,24,48,224,224
1380 DATA 6,4,12,56,224,128,128,224
1390 DATA 32,48,24,8,8,8,9,15
1400 :
1410 DATA 0,49,49,49,63,32,32,32
1420 DATA 0,140,140,140,252,4,100,103
1430 DATA 32,33,48,24,12,7,3,2
1440 DATA 4,132,252,24,48,224,192,64
1450 DATA 2,2,2,30,240,192,96,32
1460 DATA 64,96,32,32,32,96,64,120
1470 :
1480 :
1490 REM ** INIZIALIZZA LE STRINGHE DELLE IMMAGINI
1500 :
1510 IMAGE$(0) = "AB[illegible]CD[illegible]EF[illegible]"
1520 IMAGE$(1) = "GH[illegible]IJ[illegible]KL[illegible]"
1530 IMAGE$(2) = "MN[illegible]OP[illegible]QR[illegible]"
1540 IMAGE$(3) = "ST[illegible]UV[illegible]WX[illegible]"
1550 :
1560 :
1570 REM ** PULISCE LO SCHERMO, CENTRA
1575 REM     E MODIFICA I PUNTATORI
1580 REM     AI DATI DEI  CARATTERI
1590 :
1600 PRINT "[illegible]";
1610 PRINT "[illegible]";
1620 :
```

```
1630 VIC = 53248
1640 POKE (VIC+24),29
1650 :
1660 :
1670 REM ** CAMMINA
1680 :
1690 FOR N = 0 TO 3
1700 :    PRINT IMAGE$(N);
1710 :    FOR DLY = 1 TO 70 : NEXT DLY
1720 :    GET KP$
1730 :    IF KP$ = "" THEN 1750
1740 :       KEY = -1 : N = 3
1750 NEXT N
1760 :
1770 IF (NOT KEY) THEN 1690
1780 :
1790 :
1800 REM ** PULISCE TUTTO
1810 :
1820 PRINT "J";
1830 POKE (VIC+24),21
1840 :
1850 END
```

FIG. 4.10. Listato del programma "Alieno cammina".

L'istruzione 1100 fissa la base della nostra memoria dei caratteri alla stessa locazione usata precedentemente: 12288. Le linee 1110-1180 inizializzano un ciclo che girerà dal codice 1, che sta per la A, fino al codice 24, che sta per X. Un ciclo interno, inizializzato dalle linee 1120-1160, legge gli 8 byte dei dati per ogni carattere e poi li inserisce nella locazione esatta. La linea 1140 stabilisce la posizione giusta usando una formula simile a quella usata a pagina xx.
Le linee da 1200 fino a 1460 contengono codici dei modelli basati sulle immagini della figura 4.9. Ogni linea di dati contiene i codici per una nuova definizione di carattere.
Le linee 1510-1540 costruiscono quattro stringhe di immagine. Ognuna è composta da sei dei nostri nuovi caratteri, combinati con i movimenti del cursore, necessari per ottenere i 6 caratteri in un blocco di dimensione 2 × 3. Se non riconoscete i simboli grafici che rappresentano i vari movimenti del cursore nelle stringhe, consultate l'introduzione. Notate che i comandi del cursore sono usati in modo tale che, dopo che i pezzi dell'immagine sono stati disegnati, il cursore si riporta nella posizione iniziale.
Le linee 1600-1610 puliscono lo schermo e portano il cursore a metà schermo. Poi la linea 1640 dice al VIC-II di iniziare a prendere i modelli dei caratteri dal blocco di 2K, iniziando da 12288.
Finché non spostate la locazione della memoria del video, che è codificata nei bit 4, 5, 6, 7 di VIC + 24, potete usare la formula seguente per fare in modo che VIC + 24 indichi un nuovo blocco di caratteri della

memoria: dividete il nuovo indirizzo per 1024, sommate il risultato a 17 e inseritelo.
Il ciclo delle linee **1690-1750** stampa semplicemente le stringhe dell'immagine in successione con una pausa fra i cambiamenti di immagine. Le linee **1730-1740** costituiscono il controllo dei tasti che ci è già familiare. Se viene premuto un tasto, il programma terminerà pulendo lo schermo e inizializzando il puntatore del carattere della memoria in VIC + 24 al preesistente generatore di caratteri in ROM.

CHE COSA ABBIAMO IMPARATO

In questo capitolo abbiamo visto:

- La capacità del Commodore 64 di stampare i 512 caratteri già esistenti
- Le 1000 locazioni del video, i 1000 byte della memoria del video e i 1000 byte della memoria del colore
- L'inserimento della codifica del carattere e l'inserimento del colore rispettivamente nella memoria del video e nella memoria del colore
- Come si mettono insieme i movimenti del carattere e del cursore in stringhe che possono essere spostate lungo lo schermo
- Come il VIC-II sa dove cercare i modelli del carattere
- Come si trasferiscono i modelli dei caratteri dalla ROM alla RAM
- Come si disegnano e modificano gli insiemi di caratteri già esistenti
- Come si disegna un insieme di caratteri da usare in un ciclo di animazione

ESERCIZI

Test

1. Ci sono ....
caratteri diversi già esistenti nel generatore di caratteri ROM del Commodore 64.
2. Il Commodore 64 normalmente visualizza ....
linee di testo ognuna costituita da .... caratteri, in totale .... locazioni sul video.
3. Il Commodore 64 contiene ...... insiemi complet i di caratteri in ROM.
4. Premendo contemporaneamente i tasti **SHIFT** e **Commodore**, si commuta fra gli ....

5. Perché la faccia nel programma "Faccia che vola" è disegnata con un anello di spazi intorno?
6. I bit 1, 2, 3 del registro in VIC + 24 dicono al VIC-II la locazione del ....
7. Quali sono le due complicazioni che si riscontrano quando si copia il contenuto del generatore di caratteri dalla ROM in RAM?
8. Che forma assumerebbe il modello del carattere se gli 8 numeri di codifica fossero tutti 255?

Esercizi di programmazione

1. Cambiate il programma "Faccia che vola" in modo che un altro disegno si muova intorno allo schermo.
2. Cambiate il programma "ROM in RAM" in modo che i caratteri vengano stampati ribaltati.
3. Cambiate il panorama "Alieno cammina" in modo che tre alieni uguali camminino da un capo all'altro dello schermo.

Risposte ai test

1. 512
2. 25, 40, 1000
3. 2
4. due insiemi del carattere
5. così cancellerà ogni sua traccia quando si muove.
6. i modelli del carattere
7. (1) Il contenuto della ROM si sposta in diversi indirizzi di memoria. (2) Quando è bloccato, le operazioni di input/output non possono essere effettuate.
8. Un quadrato pieno.

Soluzioni possibili degli esercizi di programmazione

Queste soluzioni sono basate sull'aggiunta o il cambiamento di istruzioni nei programmi citati negli esercizi.

1. Caricate il programma "Faccia che vola". Poi inserite queste istruzioni:

```
1000 REM *** OMINO CHE VOLA ***
1060 F$(2) = "    O    "
1070 F$(3) = "  ─▒─    "
1080 F$(4) = "    ▒    "
1090 F$(5) = "   / |   "
1100 F$(6) = "  ─   ─  "
```

2. Caricate il programma “ROM in RAM”. Poi inserite queste istruzioni:

```
1000 REM *** ROM CAPOVOLTA ***
1270 : POKE DS+(7-BYTE),PEEK(SR+BYTE)
```

3. Caricate il programma “Alieno cammina”. Poi inserite queste istruzioni:

```
1000 REM *** 3 ALIENI IN CAMMINO ***
1610 PRINT "■■■■■■■■■■■■■";
1701 :     PRINT "■■■■■■";
1702 :    PRINT IMAGE$(N);
1703 :     PRINT "■■■■■■";
1704 :    PRINT IMAGE$(N);
1705 :     PRINT "■■■■■■■■■■■■■";
1706 :
1710 :    FOR DLY = 1 TO 60 : NEXT DLY
```

CAPITOLO CINQUE

Grafica a mappa di bit

Fino ad ora, abbiamo esplorato solo due aspetti della grafica del Commodore 64: gli sprite e i caratteri.
Entrambe le entità grafiche vi permettono di manipolare insiemi di pixel. Esiste un metodo per disegnare figure grandi e dettagliate controllando i pixel individualmente? Sì: è la grafica a mappa di bit.
In questo capitolo impareremo come realizzare il modo a mappa di bit. Accenderemo e spegneremo i pixel individualmente e assegneremo loro il colore. Vi darò una routine in linguaggio macchina che accelererà un aspetto noioso della tecnica del disegno della mappa di bit. Infine costruiremo un programma semplice ed elettronicamente sofisticato.

64.000 PIXEL

È ora di applicare un po' di aritmetica. Considerate la visualizzazione di testi del Commodore 64. Ci sono 25 linee, ognuna delle quali è costituita da 40 caratteri. Ogni carattere è largo e alto 8 pixel. Questo ci dà 8 × 40 = 320 pixel trasversali sullo schermo, e 8 × 25 = 200 pixel dall'alto al basso. I 320 pixel moltiplicati per gli altri 200 generano un totale di 64.000 pixel.
Nel modo a mappa di bit si controlla ciascun pixel con un bit. Poiché in un byte sono immagazzinati 8 bit, potete dividere 64.000 per 8 e trovare che avete bisogno di 8000 byte per controllare lo schermo su cui sono visualizzati i 64000 pixel. Questi 8000 byte formano la grafica a mappa di bit. Dove si può immagazzinare questa grande mappa di bit?

MEMORIZZAZIONE DELLA MAPPA DI BIT

Abbiamo già detto che l'integrato VIC-II controlla 16K di memoria alla volta. 8000 byte sono al più 8K, la metà di un blocco di 16K della memoria. Una mappa di bit di 8000 byte può stare nella prima o nella seconda metà del banco attivato di VIC-II.
Quando si lavora in Basic, VIC-II controlla il blocco della memoria dalla locazione 0 alla 16383. Le prime migliaia di locazioni in quel blocco sono molto importanti per il Basic, che non le abbandonerà facilmente. Allora la mappa di bit entra nella seconda metà del blocco, che inizia dalla locazione di memoria 8192. Il bit 3 del registro VIC + 24 (locazione di memoria 53272) controlla la locazione della mappa di bit. Se vi è memorizzato uno 0, entra nella prima metà del banco corrente di 16K di VIC-II. Se si inserisce un 1 nel bit 3 di VIC + 24, la mappa di bit viene posta nella seconda metà del banco di 16K, che è ciò che viene fatto normalmente quando si usa una mappa di bit dal Basic. Questo comando in Basic inserirà uno 0 nel bit 3 di VIC + 24 (53272):

```
POKE 53272, PEEK (53272) AND 247
```

E quest'altro inserirà un 1 in quella posizione:

```
POKE 53272, PEEK (53272) OR 8
```

ATTIVARE E DISATTIVARE IL MODO MAPPA DI BIT

Il bit 5 del registro in VIC + 17 (locazione di memoria 53265) controlla il modo della grafica a mappa di bit.
Inserendo un 1 in quella locazione, si attiva il modo mappa di bit, inserendo uno 0 lo si disattiva. Questo è il comando in Basic per poter attivare il modo mappa di bit:

```
POKE 53265, PEEK (53265) OR 32
```

Questo è il comando per disattivarlo, ridando una normale visualizzazione in modo testo:

```
POKE 53265, PEEK (53265) AND 223
```

UNA PICCOLA RESTRIZIONE

Il Basic è un buon linguaggio, ma presenta anche degli svantaggi. I programmi possono essere scritti e corretti abbastanza velocemente, ma girano molto lentamente rispetto a programmi scritti in altri linguaggi. Ovviamente, in molte applicazioni i problemi di velocità del Basic non si notano e la sua facilità di utilizzo è un aiuto notevole.
Il problema della velocità sorge per quei programmi dove molte attività sono ripetute. La grafica a mappa di bit dove 64000 bit aspettano istruzioni, è una delle aree in cui maggiormente si manifesta la lentezza del Basic.
Come si può velocizzare questo tipo di programmi? La tecnica migliore è stendere i programmi in modo intelligente. Per esempio, molti calcoli possono essere eseguiti una sola volta, inserendo i risultati in tabelle di dati, anziché essere ripetuti continuamente. L'abilità che acquisterete cercando di applicare intelligenti tecniche di stesura dei programmi, potrete trasferirla ad altri linguaggi di programmazione.
Una tecnica molto usata, che però non mi soddisfa molto, induce a compattare i codici con tante istruzioni in una linea, quante lo spazio ne permette. Io trovo che il risparmio di tempo che si ottiene utilizzando questa tecnica sia minimo e i problemi di messa a punto dei programmi siano deprimenti.
Una terza alternativa è quella di codificare in linguaggio macchina le operazioni più critiche. A meno di scrivere l'intero programma in linguaggio macchina, questa tecnica è la migliore per quanto riguarda il risparmio di tempo. Vedremo un esempio più avanti nel capitolo.

UN ULTIMO PARTICOLARE: IL COLORE

Prima di affrontare un esempio di programma, occorre discutere un ultimo particolare: il colore.
Come decide VIC-II il colore da attribuire a ognuno dei 64.000 pixel? Con la normale grafica a mappa di bit, i pixel in ogni sezione dello schermo di 8 × 8 pixel hanno una scelta fra due colori.
Il fatto che queste aree abbiano la stessa dimensione di un carattere nel modo testo porta a un'idea intelligente per quanto riguarda l'immagazzinamento. La codifica dei 2 colori per ogni area di 8 × 8 è immagazzinata nelle 1000 locazioni di memoria dello schermo, che è la stessa area usata nel modo testo per contenere i codici di visualizzazione sullo schermo.

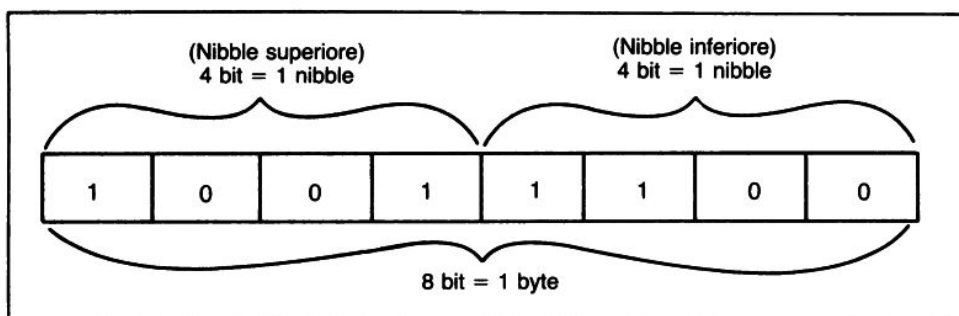



FIG. 5.1. Relazione fra bit, byte e nibble.

Le persone che lavorano col computer preferiscono nomi brevi. 8 bit costituiscono un byte e 4 bit sono un nibble.
Guardate la figura 5.1. Un nibble può immagazzinare valori fra 0 e 15.
Guardate la figura 5.2. Nel modo mappa di bit, i 4 bit superiori di ogni locazione della memoria del video contengono la codifica per qualsiasi bit che contiene il valore 1 nell'area 8 × 8 controllata da quella locazione

FIG. 5.2. Alcuni nibble tipici con i corrispondenti valori in base 10.

di memoria. Il nibble inferiore della locazione della memoria dello schermo contiene la codifica del colore per i bit contenenti il valore 0. Guardate la figura 5.3, per esempio. C'è una piccola formula per aiutarvi a scoprire quali numeri inserire in questa memoria del video per una data coppia di colori: prendete il codice del colore per i bit che contengono 1, moltiplicatela per 16 e poi sommate la codifica del colore e i bit che contengono 0. Per esempio, se aveste voluto colorare di rosso i bit che contengono 1 (codice di colore 2) e colorare di nero i bit che hanno valore 0 (codice di colore 0) avreste dovuto calcolare $(2 \times 16) + 0 = 32$ e inserire questo valore nella memoria dello schermo.

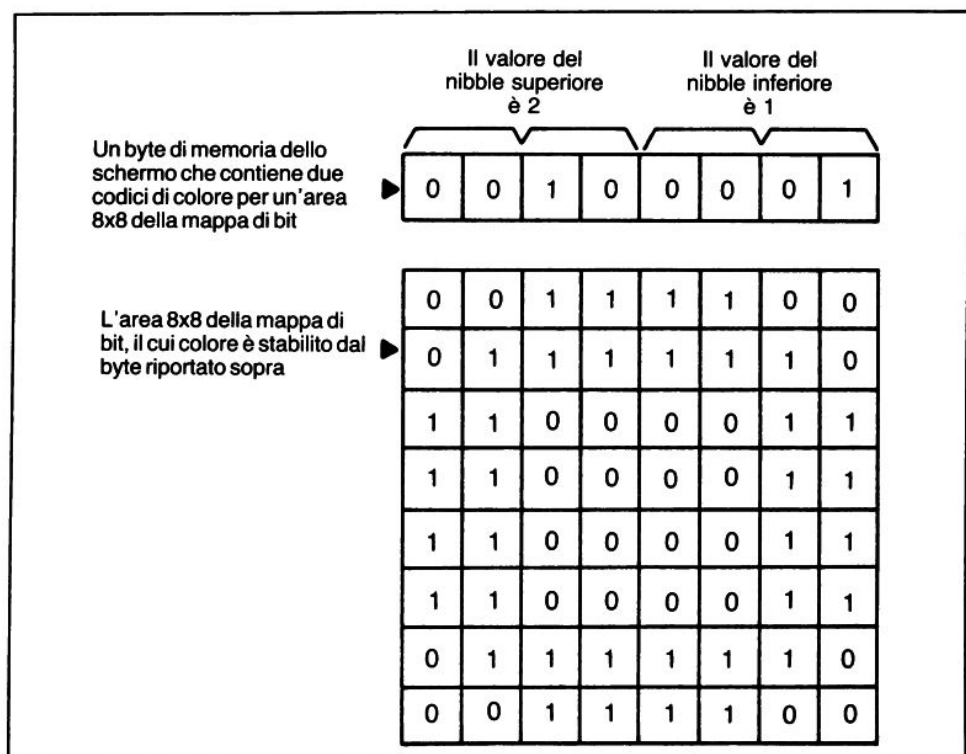



FIG. 5.3. Un esempio di area di 8x8 bit della mappa di bit il cui colore è controllato da un byte della memoria del video. Il valore del nibble superiore del byte codifica il colore per i bit nella mappa ai quali è attribuito il valore 1, mentre il valore del nibble inferiore codifica il colore per i bit nella mappa ai quali è attribuito il valore 0.

UN ESEMPIO DI GRAFICA A MAPPA DI BIT

Dopo tanta teoria, è ora di passare ad alcune applicazioni. Inserite il programma in figura 5.4, "Disegno casuale", salvatelo su nastro o su

```
1000 REM *** DISEGNO CASUALE ***
1010 :
1020 :
1030 REM ** PREPARA IL MODO MAPPA DI BIT (MMB)
1040 :
1050 VIC = 53248
1060 BASE = 8192    :REM INIZIO MAPPA DI BIT
1070 BLOC = VIC+24  :REM LOCALIZZA LA MAPPA DI BIT
1080 BSET = VIC+17  :REM ATTIVA MMB
1090 :
1100 POKE BLOC, PEEK(BLOC) OR 8
1110 POKE BSET, PEEK(BSET) OR 32
1120 :
1130 :
1140 REM ** PULISCE LA MAPPA DI BIT
1150 :
1160 FOR SPOT = BASE TO BASE + 7999
1170 :     POKE SPOT, 0
1180 NEXT SPOT
1190 :
1200 :
1210 REM ** SEME DELLA FUNZIONE CASUALE
1220 REM    CON UN NUMERO CASUALE
1230 :
1240 DUMMY = RND (-RND(0))
1250 :
1260 :
1270 REM ** FISSA I COLORI DELLA MAPPA
1280 REM    DI BIT A CASO
1290 :
1300 FOR SPOT = 1024 TO 2023
1310 :     POKE SPOT, INT (RND(1) * 256)
1320 NEXT SPOT
1330 :
1340 :
1350 REM ** DISEGNA A CASO FINO A CHE LA
1360 REM    PRESSIONE DI UN TASTO INTERROMPE
1370 :
1380 SPOT = INT (RND(1) * 8000) + BASE
1390 PATTERN = INT (RND(1) * 256)
1400 POKE SPOT, PATTERN
1410 :
1420 GET KP$
1430 IF KP$ = "" THEN 1380
1440 :
1450 :
1460 REM ** PULISCE TUTTO
1470 :
1480 POKE BSET, PEEK(BSET) AND 223
1490 POKE BLOC, 21
1500 :
1510 END
```

FIG. 5.4. Listato del programma "Disegno casuale".

disco e fatelo girare; guardatelo per un paio di minuti e poi lasciatelo girare da solo per 5 o 10 minuti.
Date un ultimo sguardo e premete la barra spazio per farlo terminare.

Come entrare nel modo mappa di bit

Esaminate il programma e guardate se riuscite a capire quello che è successo sul video. La linea 1100 usa il comando discusso a pagina per localizzare la mappa di bit nelle locazioni 8192-16191. Poi la linea 1110 attiva il modo mappa di bit con i comandi visti a pagina . La visualizzazione cambia immediatamente. Vedrete uno schermo che unisce quadrati colorati con piccole forme irregolari rosse e nere.
VIC-II sta ora cercando di interpretare le locazioni di memoria 8192-16191 come una mappa di bit contenente informazioni di visualizzazione dei pixel.
Poiché non abbiamo inserito alcuna configurazione in questa area di memoria, il video mostra gruppi di punti e linee apparentemente irregolari. La memoria del video, che contiene ancora i codici di visualizzazione dei caratteri in modo testo, è stata interpretata come informazione di colore dei pixel. Dove nessun carattere era visibile, era stata inserita la codifica 32 per uno spazio.
Questo numero inserisce pixel rossi e neri nelle corrispondenti aree di 8 × 8. Dove si vedevano i caratteri, la varietà dei codici produce una gamma di combinazioni di colore.

Come si pulisce la mappa di bit

Le istruzioni 1160-1180 puliscono la mappa di bit inizializzando a 0 tutte le locazioni di memoria della mappa:

```
1160 FOR SPOT = BASE TO BASE + 7999
1170    POKE SPOT, 0
1180 NEXT SPOT
```

Con i bit a 0 tutti i pixel in ogni area 8 × 8 verranno visualizzati nel colore codificato nel nibble inferiore di quel byte dell'area di memoria dello schermo; i disegni irregolari scompaiono piano piano, perché il Basic deve inserire uno 0 in 8000 locazioni di memoria.

Assegnare colori a caso

Il Basic del Commodore ha una buona funzione di generazione di numeri casuali. Lo strano codice nella linea 1240 prende un numero dall'orologio del sistema e lo usa come "seme" per il generatore di numeri casuali. In questo modo si possono ottenere numeri casuali diversi ogni volta che il programma viene fatto girare.
Le linee 1300-1320 riempiono l'area di memoria del video con valori casuali compresi fra 0 e 255:

```
1300 FOR SPOT = 1024 TO 2023
1310 :    POKE SPOT, INT (RND(1) * 256)
1320 NEXT SPOT
```

Il video si riempie velocemente di una varietà di blocchi colorati. Questo perché molti valori diversi stanno ora riempiendo i nibble inferiori dell'area delle informazioni sul colore. La linea 1310 è un punto interessante su cui fermarsi e sperimentare diverse formule di determinazione del colore. Provate per esempio:

```
1310 :    POKE SPOT, INT(RND(1)*16) * 16
```

Questo è un modo piacevole per imparare alcune cose sui numeri e sulle funzioni.

Disegnare forme casuali in punti casuali

Le linee 1380-1400 continuano questa tendenza casuale del programma:

```
1380 SPOT = INT (RND(1) * 8000) + BASE
1390 PATTERN = INT (RND(1) * 256)
1400 POKE SPOT, PATTERN
```

La linea 1380 prende a caso un punto nella mappa di bit. Poi la linea 1390 prende a caso una sequenza di bit. La linea 1400 inserisce la sequenza nel punto. Questo processo di disegno viene ripetuto finché non si preme un tasto e quindi il programma termina. La linea 1480 disattiva il modo mappa di bit e la linea 1490 porta di nuovo VIC + 24 a puntare all'insieme di caratteri interno. Tutto questo, grazie alla doppia personalità del bit 3 di VIC + 24. Quando è attivo, il modo mappa di bit controlla la locazione della mappa di bit; nel modo testo è uno dei 3 bit che collocano le configurazioni dei caratteri.

Le linee **1380-1400** sono un altro punto eccellente per la sperimentazione. Cambiando le formule in quelle righe, potete produrre figure interessanti. Dopo aver sperimentato un po', divertitevi un po' a creare forme nuove, combinate questi cambiamenti con altri cambiamenti fatti in altre parti del programma.

UNA SCORCIATOIA

Il paragrafo sul modo di pulire la mappa di bit è uno dei più noiosi, per quanto riguarda il "Disegno casuale". Non vi è modo di velocizzare il processo, finché si utilizza il Basic. Non si può tralasciare l'assegnamento dei valori nelle 8000 locazioni di memoria.
Questo è un lavoro per il linguaggio macchina!
Il linguaggio macchina è l'unico linguaggio che il vostro Commodore 64 capisce effettivamente; il Basic è lento perché ogni istruzione deve essere tradotta in linguaggio macchina. La figura 5.5 elenca alcuni cambiamenti e istruzioni ulteriori che potete inserire in "Disegno casuale" per inserirvi una veloce subroutine in linguaggio macchina per pulire la mappa di bit. Caricate "Disegno casuale", inserite le nuove istruzioni, salvate la nuova versione e poi fatelo girare. Potete capire ora il motivo per cui i programmatori si rivolgono al linguaggio macchina quando hanno bisogno di velocità.
Una breve spiegazione delle nuove istruzioni: le linee **1146-1156** inseri-

```
1000 REM *** DISEGNO CASUALE VELOCE ***
1140 REM ** CARICA LA ROUTINE VELOCE IN
1141 REM    LINGUAGGIO MACCHINA PER LA
1142 REM    PULIZIA DELLA MAPPA DI BIT
1143 :
1146 FOR N = 21248 TO 21273
1150 :   READ MLDTA
1153 :   POKE N, MLDTA
1156 NEXT N
1160 :
1163 DATA 169, 0, 133, 251, 169, 32
1166 DATA 133, 252, 162, 32, 160, 0
1170 DATA 152, 145, 251, 200, 208, 251
1173 DATA 202, 240, 4, 230, 252, 208
1176 DATA 244, 96
1180 :
1183 :
1186 REM ** PULISCE LA MAPPA DI BIT
1190 :
1193 SYS 21248
1196 :
```

FIG. 5.5. Cambiamenti e istruzioni addizionali che trasformano il programma "Disegno casuale" nel programma "Disegno casuale veloce".

sono la subroutine in linguaggio macchina in un'area di memoria che la maggior parte dei programmi in Basic non usano. Le linee 1163-1176 contengono i 26 byte di dati che costituiscono il procedimento di cancellazione. Infine la linea 1193 chiama in azione la subroutine in linguaggio macchina appena installata con un comando SYS. È come saltare a una subroutine in Basic. Quando la subroutine in linguaggio macchina termina, lascia il controllo al Basic e quest'ultimo continua con nuove istruzioni.
Potete usare questa routine in qualsiasi programma di disegno a mappa di bit che usi le locazioni 8192-16191 come area della mappa di bit. Se volete pulire una mappa di bit che inizia in un'altra area, dividete l'indirizzo iniziale della mappa per 256 e inserite il nuovo valore al posto di 32 alla fine della linea 1163.

LOCAZIONE DEL BYTE E DEL BIT DI UN PIXEL

Impariamo a ottenere un maggior controllo sui singoli pixel nel modo mappa di bit. Abbiamo bisogno di trovare un modo per immagazzinare il byte e il bit che controllano il singolo pixel.
Primo, avete bisogno di un modello dello schermo. Guardate la figura 5.6. Ogni pixel ha una posizione orizzontale, H, con valori da 0 a 319. Ogni pixel ha inoltre una posizione verticale, V, con valori da 0 a 199. Per esempio, un pixel che si trova nell'angolo a sinistra in alto ha H = 0 e V = 0. Un pixel nell'angolo a destra in basso ha H = 319 e V = 199. Sarebbe stupendo se i byte nella mappa di bit avessero una semplice corrispondenza con la figura 5.6.

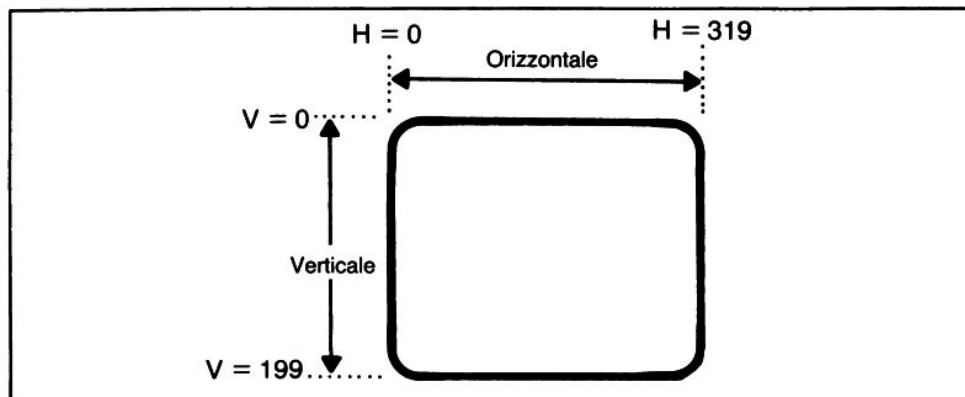



FIG. 5.6. Potete attribuire a ogni pixel nel disegno del bit una posizione orizzontale da 0 a 319 e una posizione verticale da 0 a 99.

Sfortunatamente, ciò non avviene. Il byte nella mappa di bit corrisponde allo schermo in un modello che suggerisce la stretta parentela della mappa di bit con la visualizzazione nel modo testo.
Guardate la figura 5.7. Mostra come vengono inizializzati i byte della mappa di bit. Gruppi di 8 byte consecutivi formano un blocco che ha le dimensioni di un carattere.
Come nel modo testo, queste aree alte 8 byte sono disposte in 40 colonne e 25 righe. Cercare di determinare quale bit di quale byte controlla un pixel, data la posizione orizzontale e verticale del pixel,

	Colonna 0	Colonna 1	Colonna 2		Colonna 39
Riga 0	Byte 0	Byte 8	Byte 16		Byte 312
	Byte 1	Byte 9	Byte 17		Byte 313
	Byte 2	Byte 10	Byte 18		Byte 314
	Byte 3	Byte 11	Byte 19		Byte 315
	Byte 4	Byte 12	Byte 20	...	Byte 316
	Byte 5	Byte 13	Byte 21		Byte 317
	Byte 6	Byte 14	Byte 22		Byte 318
	Byte 7	Byte 15	Byte 23		Byte 319
Riga 1	Byte 320	Byte 328	Byte 336		Byte 632
	Byte 321	Byte 329	Byte 337		Byte 633
	Byte 322	Byte 330	Byte 338		Byte 634
	Byte 323	Byte 331	Byte 339	...	Byte 635
	Byte 324	Byte 332	Byte 340		Byte 636
	Byte 325	Byte 333	Byte 341		Byte 637
	Byte 326	Byte 334	Byte 342		Byte 638
	Byte 327	Byte 335	Byte 343		Byte 639
	⋮	⋮	⋮		⋮
Riga 24	Byte 7680	Byte 7688	Byte 7696		Byte 7992
	Byte 7681	Byte 7689	Byte 7697		Byte 7993
	Byte 7682	Byte 7690	Byte 7698		Byte 7994
	Byte 7683	Byte 7691	Byte 7699		Byte 7995
	Byte 7684	Byte 7692	Byte 7700	...	Byte 7996
	Byte 7685	Byte 7693	Byte 7701		Byte 7997
	Byte 7686	Byte 7694	Byte 7702		Byte 7998
	Byte 7687	Byte 7695	Byte 7703		Byte 7999

FIG. 5.7. Come sono fissati i byte della mappa di bit. Notate la stretta relazione con la visualizzazione in modo testo del Commodore 64.

sembrerebbe una impresa assurda. In realtà, però non lo è. Se andate piano e riguardate le figure 5.6 e 5.7, potrete dare un senso alle formule che seguono. Ricordate, H e V sono riferite rispettivamente alle posizioni orizzontale e verticale di un pixel.
Iniziamo con una informazione verticale. Poiché una riga è alta 8 posizioni verticali, questa formula ci dà la riga che contiene un pixel:

```
RIGA = INT(V/8)
```

Ci sono 320 byte per riga, così lo spostamento in byte dalla base della mappa di bit è:

```
RBF = RIGA * 320
```

La funzione **AND** è un modo comodo per trovare i resti quando si divide per una potenza di 2: semplicemente si calcola l'**AND** del numero originale con il divisore meno 1. Il resto della posizione verticale diviso per 8 ci dirà quale vogliamo delle 8 linee in una riga:

```
LINEA = (V AND 7)
```

Potete combinare questi risultati, per avere uno spostamento totale verticale del byte per il pixel:

```
VBF = INT(V/8) * 320 + (V AND 7)
```

Ora dovete lavorare con la posizione orizzontale del pixel. Ci sono 8 posizioni orizzontali per colonna, così la colonna può essere rappresentata in questo modo:

```
COLONNA = INT(H/8)
```

Notate come vi muovete di colonna in colonna: c'è un salto di 8 byte. Ora rappresentate il vostro fattore di spostamento totale orizzontale del byte:

```
HBF = INT(H/8) * 8
```

Ora potete sommare gli spostamenti orizzontale e verticale del byte rispetto all'inizio della mappa di bit per arrivare al byte cercato:

```
BYTE = BASE + VBF + HBF
```

Avete il byte, dovete trovare il bit. Ci sono 8 pixel in una colonna, dovete sapere quanti pixel avanzano dopo aver considerato tutte le colonne piene. Utilizzate ancora un'operazione AND per trovare il resto:

PXL = (H AND 7)

Poiché i bit in un byte sono numerati da destra a sinistra e le vostre posizioni orizzontali del pixel vanno da sinistra a destra, dovete fare un piccolo aggiustamento, con un'operazione di inversione:

BIT = 7 − (H AND 7)

Così ora avete delle formule per calcolare un byte di un pixel del disegno del bit ed un bit. Utilizziamo un po' questi byte e questi bit.

ACCENDERE E SPEGNERE I PIXEL

Una volta trovato il byte e il bit del pixel con le formule sviluppate nel paragrafo precedente, l'enunciato seguente attribuirà al bit il valore 1:

```
POKE BYTE, PEEK(BYTE) OR (2↑BIT)
```

Ricordate: questo enunciato dirà al pixel di assumere un colore il cui codice si trova nel nibble superiore di un byte della memoria dello schermo.
Questo comando attribuirà al bit del pixel il valore 0:

```
POKE BYTE, PEEK(BYTE) AND (255 - 2↑BIT)
```

Il pixel assumerà allora il colore il cui codice è nel nibble inferiore dell'appropriato byte della memoria del video.

IL GHIRIGORO ELETTRONICO

Ora che potete accendere e spegnere i pixel singolarmente, giochiamo un po' con un programma di scarabocchi elettronici. La figura 5.8 mostra il listato del programma "Schizzo".
Inseritelo, salvatelo e fatelo girare.

```
1000 REM *** SCHIZZO ***
1010 :
1020 :
1030 REM ** PREPARAZIONE INIZIALE
1040 :
1050 PRINT "J";       :REM PULISCE LO SCHERMO
1060 POKE 650, 128  :REM TUTTI I TASTI RIPETONO
1070 :
1080 BASE = 8192    :REM INIZIO MAPPA DI BIT
1090 VIC = 53248    :REM CHIP GRAFICO
1100 BLOC = VIC+24  :REM FISSA LA BASE
1110 BSET = VIC+17  :REM ATTIVA MMB
1120 :
1130 :
1140 REM ** CARICA LA ROUTINE VELOCE
1141 REM    DI PULIZIA IN LINGUAGGIO MACCHINA
1150 :
1160 FOR N = 21248 TO 21273
1170 :    READ MLDTA
1180 :    POKE N, MLDTA
1190 NEXT N
1200 :
1210 DATA 169, 0, 133, 251, 169, 32
1220 DATA 133, 252, 162, 32, 160, 0
1230 DATA 152, 145, 251, 200, 208, 251
1240 DATA 202, 240, 4, 230, 252, 208
1250 DATA 244, 96
1260 :
1270 :
1280 REM ** PREPARA PER MMP, PULISCE LA
1290 REM     MAPPA, FISSA L'INSIEME DEI COLORI
1300 :
1310 POKE BLOC, PEEK(BLOC) OR 8
1320 POKE BSET, PEEK(BSET) OR 32
1330 :
1340 SYS 21248     :REM PULIZIA MAPPA IN L.M.
1350 :
1360 FOR HUEMAP = 1024 TO 2023
1370 :    POKE HUEMAP, 3
1380 NEXT HUEMAP
1390 :
1400 :
1410 REM ** INIZIALIZZA H E V
1420 :
1430 H = 160 : V = 100
1440 :
1450 :
1460 REM ** DISEGNA IL PUNTO A H,V
1470 :
1480 VBF = INT (V/8) * 320 + (V AND 7)
1490 HBF = INT (H/8) * 8
1500 BIT = 7 - (H AND 7)
1510 BYTE = BASE + VBF + HBF
1520 POKE BYTE, PEEK(BYTE) OR (2↑BIT)
1530 :
1540 :
1550 REM ** PRENDI IL COMANDO DA TASTO
1560 :
1570 GET KP$
1580 IF KP$ = "" THEN 1570
1590 :
1600 :
1610 REM ** GESTIONE DEL TASTO DIGITATO
1620 :
1630 IF KP$ = " " THEN 1850
```

```
1640 IF KP$ = "9" THEN SYS 21248 : GOTO 1430
1650 :
1660 IF KP$ = "W" THEN              V=V-1
1670 IF KP$ = "E" THEN H=H+1  :  V=V-1
1680 IF KP$ = "D" THEN H=H+1
1690 IF KP$ = "C" THEN H=H+1  :  V=V+1
1700 IF KP$ = "X" THEN              V=V+1
1710 IF KP$ = "Z" THEN H=H-1  :  V=V+1
1720 IF KP$ = "A" THEN H=H-1
1730 IF KP$ = "Q" THEN H=H-1  :  V=V-1
1740 :
1750 IF V < 0   THEN V = 0
1760 IF V > 199 THEN V = 199
1770 IF H < 0   THEN H = 0
1780 IF H > 319 THEN H = 319
1790 :
1800 GOTO 1480
1810 :
1820 :
1830 REM ** AVVOLGIMENTO (WRAP)
1840 :
1850 POKE BSET, PEEK(BSET) AND 223
1860 POKE BLOC, 21
1870 :
1880 PRINT "J";
1890 :
1900 END
```

FIG. 5.8. Listato del programma "Schizzo".

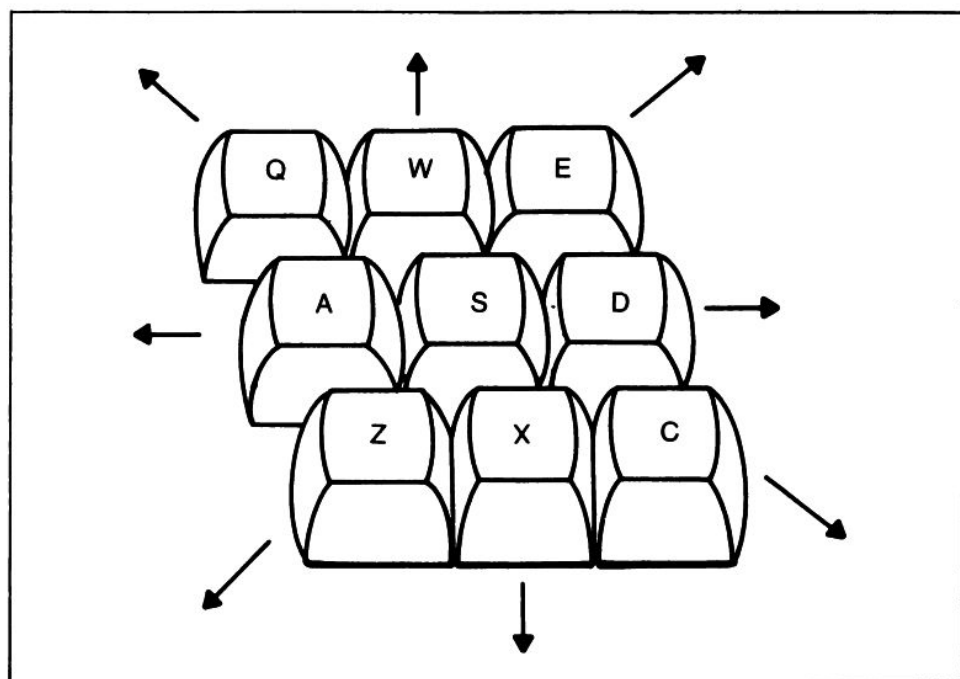



FIG. 5.9. Disposizione dei tasti di controllo usati nel programma "Schizzo" e le direzioni che verranno assegnate alla penna.

Al centro dello schermo apparirà una penna a forma di punto. Potete muovere la penna in ognuna delle 8 direzioni digitando W, E, D, C, X, Z, A, G.
La figura 5.9 mostra la disposizione di questi tasti e la direzione in cui ognuno manderà la penna. Digitando il tasto S il vostro disegno viene cancellato e la penna si riposiziona al centro dello schermo.
Quando avete finito di divertirvi, premete la sbarra spazio per porre fine al programma, poi preparatevi a una piccola spiegazione sul suo funzionamento.

Inizializzare lo Schizzo

Le linee 1000-1340 vi saranno familiari. Si pulisce lo schermo e si imposta la tastiera in modo che i tasti si ripetano quando vengono tenuti premuti per un tempo abbastanza lungo. Le linee 1160-1190 caricano la routine veloce in linguaggio macchina per pulire la mappa di bit. Poi le linee 1310-1340 inizializzano la mappa di bit. Poi le linee 1310-1340 inizializzano la mappa di bit e usano la routine in linguaggio-macchina. Le linee 1360-1380 riempiono la memoria dello schermo con uno schema di colore per la mappa di bit. I bit che hanno valore 0 diventeranno azzurri e i bit che hanno valore 1 diventeranno neri. Dato che la linea 1340 riempiva la mappa di bit di 0, lo schermo diventa azzurro. Immagazzinate le posizioni orizzontale e verticale della penna nelle variabili H e V. La linea 1430 inizializza queste variabili, portando la penna al centro dello schermo. Quando si digiterà il tasto S, il programma ritornerà a questa linea.

Disegnare

Le linee 1480-1520 usano le formule sviluppate nelle pagine precedenti per accendere il bit corrispondente alla posizione attuale della penna. Inserendo un 1 nel bit il pixel nella posizione della penna diventa nero.

Ricevere ed eseguire gli ordini

Le linee 1570-1580 aspettano che il disegnatore prema un tasto. Poi le linee 1630-1800 stabiliscono che cosa succede quando viene premuto un tasto. Uno spazio manda il programma alla linea 1850, che azzera tutto e fa terminare il programma. Digitando S la mappa di bit viene pulita e la penna torna al centro; il controllo torna alla linea 1430.

Le linee 1660-1730 cambiano la posizione della penna se viene premuto uno degli 8 tasti.
Le linee 1750-1780 controllano che la penna non cada fuori dallo schermo. Se un tasto cerca di farla uscire queste quattro linee la reinseriscono. Infine la linea 1800 ritorna ciclicamente a disegnare il punto della penna sullo schermo.
Notate che qualsiasi tasto non incluso nei comandi del programma sarà ignorato. Inoltre, la struttura chiara di questa sezione rende facile l'aggiunta di nuovi comandi.
Le linee 1850-1900 determinano la fine del programma. Riportano la visualizzazione al modo testo e puliscono lo schermo. È lo stesso modo in cui terminava "Disegno casuale". Ora divertitevi un po' con "Schizzo", guardate quali altre caratteristiche potete aggiungergli.

CHE COSA ABBIAMO IMPARATO

Questo capitolo ha introdotto alcune tecniche per la grafica a mappa di bit. In particolare, ora sappiamo:

- Come rappresentare 64000 pixel di schermo in una mappa di bit di 8000 byte
- Dove di solito si immagazzina la mappa di bit quando si lavora in Basic e come dire a VIC-II la locazione
- Come attivare e disattivare il modo mappa di bit tramite il registro VIC + 17
- Perché per una grafica a mappa di bit davvero veloce si ricorre all'uso delle routine in linguaggio macchina
- Come è usata la memoria del video per fornire l'informazione del colore per i pixel nel modo mappa di bit
- Alcuni dei modi con cui possono essere usati i numeri casuali per creare grafica a mappa di bit
- Come trovare il byte ed il bit che controllano un singolo pixel in un disegno a mappa di bit
- Come assegnare a un singolo pixel uno dei due colori possibili nel suo blocco

A questo punto conosciamo le tre principali capacità grafiche del Commodore: sprite, grafica a caratteri e grafica a mappa di bit.
Nel prossimo capitolo vedremo altri trucchi per la grafica.

Test

1. La grafica a mappa di bit dà la possibilità di controllare .... pixel dello schermo con un disegno a mappa di .... byte.
2. Quando si usa il Basic, la mappa di bit di solito è collocata nella .... metà dei primi 16K della memoria.
3. Il bit 5 del registro di .... (locazione di memoria 53265) attiva e disattiva il modo mappa di bit.
4. Perché sono spesso usate le routine in linguaggio-macchina per la grafica a mappa di bit?
5. Nel modo mappa di bit, i 2 nibble di un byte della memoria dello schermo sono usati per ....
6. Quali linee del programma "Disegno casuale" assegnano i colori per la mappa di bit?
7. Il comando .... dà la possibilità di passare dal Basic a una subroutine in linguaggio macchina.
8. La relazione fra byte nella mappa di bit e pixel sullo schermo è ....
9. Assegnare a un bit nella mappa di bit il valore 1, attribuisce al relativo pixel il colore che si trova nel nibble .... di un byte nella memoria dello schermo.
10. Che cosa succederebbe al programma "Schizzo" se la linea 1640 saltasse alla linea 1480 invece che alla 1430?

Esercizi di programmazione

1. Cambiate il programma "Disegno casuale" affinché disegni linee verticali colorate a caso su uno sfondo nero.
2. Cambiate il programma "Schizzo" affinché formi linee due volte più larghe. Avvertenza: il programma sarà probabilmente lento. È un tipico caso in cui sarebbe necessaria una nuova stesura del programma e/o l'introduzione di routine in linguaggio macchina.
3. Ecco un esercizio a prima vista difficile, ma davvero non male. Si possono usare gli sprite nella grafica a mappa di bit. Disegnate uno sprite che sembri una penna, una matita o un pennello. Poi cambiate il programma "Schizzo" in modo che sembri che lo sprite stia disegnando le linee.

Risposte al test

Le risposte possono variare, specialmente per le domande 4 e 8.

1. 64.000; 8000
2. Secondo
3. VIC + 17
4. Velocità
5. Attribuire i colori a un'area di un pixel 8 × 8 della visualizzazione sullo schermo
6. Le linee 1300-1320
7. SYS
8. Arcana e strana, tuttavia spesso utile
9. Superiore
10. Dopo aver cancellato il disegno, la penna non si sarebbe riposizionata al centro dello schermo ma sarebbe rimasta nella sua posizione finale

Possibili soluzioni ai problemi di programmazione

Ancora una volta queste soluzioni si basano su aggiunte e cambiamenti nei programmi citati negli esercizi.

1. Caricate il programma "Disegno casuale" e inserite queste linee:

```
1000 REM *** DISEGNO CASUALE VERTICALE ***
1310 :  POKE SPOT, INT (RND(1)*16) * 16
1380 SPOT = INT(RND(1)*1000) * 8 + BASE
1385 PATTERN = 56
1390 :
1395 FOR BYTE = 0 TO 7
1400 :   POKE SPOT + BYTE, PATTERN
1405 NEXT BYTE
```

2. Caricate il programma "Schizzo" e inserite queste linee:

```
1000 REM *** SCHIZZO GRASSO ***
1473 FOR X = H TO (H + 1)
1476 :   FOR Y = V TO (V + 1)
1480 :   VBF = INT (Y/8) * 320 + (Y AND 7)
1490 :   HBF = INT (X/8) * 8
1500 :   BIT = 7 - (X AND 7)
1510 :   BYTE = BASE + VBF + HBF
1520 :   POKE BYTE, PEEK(BYTE) OR (2↑BIT)
1523 :   NEXT Y
1526 NEXT X
1760 IF V > 198 THEN V = 198
1780 IF H > 318 THEN H = 318
1800 GOTO 1473
```

3. Caricate il programma "Schizzo" e inserite queste linee:

```
1000 REM *** SCHIZZO A MATITA ***
1251 :
1252 :
1253 REM ** CARICA I DATI DEGLI SPRITE
1254 :
1255 FOR N = 896 TO 958
1256 :   READ SPDTA
1257 :   POKE N, SPDTA
1258 NEXT N
1259 :
1260 DATA   0,   1, 224,   0,   3,  48
1261 DATA   0,   6,  24,   0,  12,  12
1262 DATA   0,  24,   6,   0,  48,   2
1263 DATA   0,  96,   6,   0, 192,  12
1264 DATA   1, 128,  24,   3,   0,  48
1265 DATA   6,   0,  96,   7,   0, 192
1266 DATA  13, 129, 128,  24, 195,   0
1267 DATA  16, 102,   0,  16,  60,   0
1268 DATA  48,  48,   0,  56, 240,   0
1269 DATA 127, 128,   0, 120,   0,   0
1270 DATA 192,   0,   0
1271 :
1272 :
1391 :
1392 REM ** FISSA I CONTROLLI DEGLI SPRITE
1393 :
1394 POKE 2040, 14  :REM FISSA PUNTATORE SPRITE 0
1395 POKE VIC+39, 0 :REM DIPINGILO DI NERO
1396 POKE VIC+29, 1 :REM ESPANSIONE ORIZZONTALE
1397 POKE VIC+23, 1 :REM ESPANSIONE VERTICALE
1398 POKE VIC, 184  :REM INIZIALIZ. POS ORIZZONTALE
1399 POKE VIC+1,109 :REM INIZIALIZ. POS VERTICALE
1400 POKE VIC+21,1  :REM ATTIVA SPRITE 0
1401 :
1402 :
1531 REM ** SPOSTA LO SPRITE
1532 :
1533 SH = H + 24 : SV = V + 9
1534 RS = (SH > 255)
1535 POKE VIC, SH + (RS * 256)
1536 POKE VIC+16, -RS
1537 POKE VIC+1, SV
1538 :
1539 :
1871 POKE VIC+21, 0  :REM DISATTIVA SPRITE 0
1872 POKE VIC+23, 0  :REM DISATTIVA ESPANSIONE
1873 POKE VIC+29, 0
```

CAPITOLO SEI

Altri trucchi per la grafica

Questo capitolo sarà un po' diverso dai precedenti. Toccheremo non approfonditamente un grande numero di caratteristiche grafiche. Le discussioni dei programmi verranno snellite, così potremo trattare più argomenti.
Questi i settori che guarderemo: far scorrere gli sprite sopra e sotto i grafici che faranno da sfondo, mettere un testo nella grafica a mappa di bit, far volare uno sprite con un joystick, identificare collisioni tra sprite e altri oggetti, inserire due altri modi di colore per la grafica a caratteri e la mappa di bit multicolore. Dobbiamo occuparci di molti argomenti, quindi passiamo al loro trattamento.

PRIORITÀ DELLO SPRITE RISPETTO ALLO SFONDO

Nel capitolo tre avevamo discusso le priorità di uno sprite rispetto ad altri sprite. Quando due o più sprite si sovrappongono sullo schermo gli sprite con numeri bassi hanno priorità di visualizzazione superiori. Per esempio lo sprite 3 apparirà davanti allo sprite 5. C'è un registro in VIC + 27 (locazione di memoria 53275) che controlla la priorità degli sprite rispetto allo sfondo. Per sfondo si intende qualsiasi figura che non faccia parte di uno sprite: caratteri e immagini in grafica a mappa di bit. Ogni sprite ha associato un bit nel registro in VIC + 27. Il bit 0 controlla lo sprite 0, il bit 1 lo sprite 1, e così via.
Se il bit di uno sprite è 1, esso ha una priorità inferiore rispetto a qualsiasi sfondo in cui è presente. Lo sprite sembrerà nascondersi dietro

FIG. 6.1. Questa regolazione del registro di controllo del rapporto fra sprite e sfondo significa che gli sprite 1, 2, 5 e 6 appariranno davanti a immagini di sfondo, mentre gli altri appariranno dietro.

lo sfondo. Se il bit di uno sprite è 0, lo sprite ha priorità superiore rispetto allo sfondo, quindi passerà davanti allo sfondo.
Guardate la figura 6.1; essa mostra una regolazione del registro di controllo dello sprite rispetto allo sfondo. Per attribuire le priorità fra sprite e sfondo si inserisce 1 nelle posizioni dei bit corrispondenti agli sprite che si vuole abbiano priorità inferiore. Poi si sommano i valori di quei bit e si inserisce il risultato in VIC + 27. La figura 6.2 è un listato del programma "Sopra e sotto". Esso usa il cambiamento di priorità per mostrare uno sprite che orbita attorno a un blocco di testo.
Inserite il programma, salvatelo e poi fatelo girare. Premendo la barra spazio il programma terminerà. Come già detto, questo capitolo darà brevi spiegazioni dei programmi: in questo modo possiamo introdurre più argomenti.
Diamo uno sguardo al programma "Sopra e sotto"; il primo modulo disegna un quadrato grande usando gli spazi azzurri con modalità reverse e i comandi che muovono il cursore. Il modulo successivo inizializza uno sprite di colore grigio medio. Successivamente si ha il modulo del programma principale. Le linee 1360-1410 muovono lo sprite a destra o a sinistra, secondo il valore della variabile di direzione DR. Digitare un tasto durante il movimento provoca la conclusione del programma, ponendo il flag BYEBYE a TRUE. Dopo un insieme di movimenti, vengono cambiate la direzione e la priorità dello sprite sullo sfondo. È sorprendente il modo in cui il semplice cambiamento di priorità possa modificare la nostra percezione del movimento dello sprite. Sembra che lo sprite stia orbitando attorno al quadrato centrale, anziché muovendosi da lato a lato.

```
1000 REM *** SOPRA E SOTTO ***
1010 :
1020 :
1030 REM ** DISEGNA LA FORMA CENTRALE
1040 :
1050 PRINT "▚";      :REM PULISCE E CENTRA
1060 PRINT "▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮";
1070 PRINT "▮▮▮▮▮▮▮▮▮▮";
1080 :
1090 PRINT "◣";      :REM DISEGNA IN CYAN
1100 FOR N = 1 TO 6
1110 :    PRINT "▮          ▮▮▮▮▮▮▮▮▮▮▮";
1120 NEXT N
1130 PRINT "▮";      :REM TORNA AL BIANCO
1140 :
1150 :
1160 REM ** INIZIALIZZA LO SPRITE
1170 :
1180 FOR N = 832 TO 894 :REM CARICA DATI
1190 :    POKE N, 255
1200 NEXT N
1210 :
1220 VIC = 53248        :REM CHIP GRAFICO
1230 POKE VIC+33, 0     :REM SFONDO NERO
1240 POKE 2040, 13      :REM PUNTATORE DATI SPRITE 0
1250 POKE VIC+39, 12    :REM SPRITE 0 GRIGIO MEDIO
1260 POKE VIC, 104      :REM POS ORIZZ SPRITE 0
1270 POKE VIC+1, 136    :REM POS VERT SPRITE 0
1280 POKE VIC+21, 1     :REM SPRITE 0 ATTIVATO
1290 :
1300 :
1310 REM ** FA VOLARE LO SPRITE
1320 :
1330 DR = 1 :REM DIREZIONE PERCORSO SPRITE
1340 PR = 0 :REM PRIORITA' SPRITE/SFONDO
1350 :
1360 FOR MOVE = 1 TO 136      :REM FLY IT
1370 :    POKE VIC, PEEK(VIC) + DR
1380 :    GET KP$
1390 :    IF KP$ = "" THEN 1410
1400 :       MOVE = 136 : BYEBYE = -1
1410 NEXT MOVE
1420 :
1430 IF BYEBYE THEN 1540   :REM FATTO?
1440 :
1450 DR = -DR      :REM CAMBIA DIREZIONE
1460 PR = 1 - PR   :REM CAMBIA PRIORITA'
1470 :
1480 POKE VIC+27,PR   :REM INSERISCE LA PRIORITA'
1490 GOTO 1360        :REM VOLA ANCORA
1500 :
1510 :
1520 REM ** PULISCE TUTTO E FINISCE
1530 :
1540 POKE VIC+21, 0 :REM SPRITE DISATTIVATO
1550 POKE VIC+27, 0 :REM RISTABILISCE LE PRIORITA'
1560 PRINT "▚";     :REM PULISCE LO SCHERMO
1570 :
1580 END
```

FIG. 6.2. Listato del programma “Sopra e sotto”.

USARE UN TESTO CON LA VISUALIZZAZIONE A MAPPA DI BIT

Nel capitolo precedente abbiamo visto la strana corrispondenza che c'è tra i byte in una mappa di bit e la visualizzazione sullo schermo: non ha molto senso quando si vogliono disegnare delle linee. È utile invece quando si vogliono aggiungere caratteri di testo a materiale in mappa di bit. Facciamo un piccolo ripasso per capire perché.
Nel modo mappa di bit, 8 byte consecutivi della memoria controllano un'area nello schermo larga e alta 8 pixel. Ogni byte controlla una riga di questo blocco d'immagine: il primo byte controlla la riga più in alto, il secondo la riga successivamente più bassa e così via.
L'informazione del carattere è immagazzinata nello stesso formato. Otto byte consecutivi della memoria formano un carattere che è alto e largo 8 pixel. Il primo byte controlla la riga più in alto del carattere, il secondo la riga successivamente più in basso e così via. Modelli per 512 caratteri sono forniti dal generatore interno in ROM; voi potete comunque disegnarne altri.
Per porre un carattere in un'immagine a mappa di bit, dovete trasferire i suoi 8 byte in una sezione di 8 byte della mappa di bit. La figura 6.3 mostra un listato che fa proprio questo. Il programma "Testo a mappa di bit" prende i modelli del carattere dalla ROM e li pone nella mappa di bit. Diamogli uno sguardo.
La prima sezione del programma inizializza varie costanti e variabili e inoltre regola la tastiera in modo che tutti i tasti si ripetano. La sezione successiva, linee 1170-1180, commuta la visualizzazione al modo mappa di bit. I successivi due segmenti creano un dipinto di Jackson Pollack. Le linee 1230-1250 inseriscono i colori per la mappa di bit. I colori per i bit con valore 0 vengono presi a caso, mentre i bit con valore 1 saranno neri. Poi le linee 1300-1320 riempiono la mappa di bit di valori casuali. Le linee 1370-1380 aspettano che voi digitiate un tasto. Se il tasto digitato è la barra spazio, il programma salta al modulo finale e termina. Le linee 1400-1410 controllano se il tasto è una lettera, un numero o un simbolo di punteggiatura.
Il successivo modulo del programma calcola il codice di visualizzazione del tasto che è stato premuto. Poi la ROM viene trasferita nella memoria. Le linee 1590-1610 copiano gli otto byte del modello del carattere nella mappa di bit, poi la ROM dei caratteri è lasciata libera. La sezione seguente aggiorna la variabile del cursore, la quale segue il corso delle nostre posizioni nella mappa di bit e poi ritorna ciclicamente ad attendere che venga premuto un altro tasto. Se non l'avete ancora fatto, inserite il programma, salvatelo e fatelo girare.

```
1000 REM *** TESTO A MAPPA DI BIT ***
1010 :
1020 :
1030 REM ** INIZIALIZZAZIONI VARIE
1040 :
1050 PRINT "J";      :REM PULISCE SCHERMO
1060 POKE 650, 128  :REM RIPETIZ. TASTI
1070 ROM = 53248    :REM ROM CARATTERI
1080 BASE = 8192    :REM BASE MAPPA DI BIT
1090 CURSR = BASE   :REM CURSORE MAPPA DI BIT
1100 VIC = 53248    :REM CHIP GRAFICO
1110 BLOC = VIC+24  :REM LOCALIZZA MAPPA DI BIT
1120 BSET = VIC+17  :REM FISSA MMB
1130 :
1140 :
1150 REM ** ATTIVA IL MODO MAPPA DI BIT
1160 :
1170 POKE BLOC, PEEK(BLOC) OR 8
1180 POKE BSET, PEEK(BSET) OR 32
1190 :
1200 :
1210 REM ** FISSA CASUALMENTE I COLORI DELLA MAPPA
1220 :
1230 FOR SL = 1024 TO 2023
1240 :    POKE SL, INT(RND(1) * 15) + 1
1250 NEXT SL
1260 :
1270 :
1280 REM ** RIEMPIE A CASO LA MAPPA
1290 :
1300 FOR BMLOC = BASE TO BASE + 7999
1310 :    POKE BMLOC, INT(RND(1) * 256)
1320 NEXT BMLOC
1330 :
1340 :
1350 REM ** PRENDE UNA LETTERA, UN NUMERO
1351 REM     UN SEGNO DI INTERPUNZIONE
1360 :
1370 GET KP$
1380 IF KP$ = "" THEN 1370
1390 IF KP$ = " " THEN 1790
1400 IF ASC(KP$) < 32 THEN 1370
1410 IF ASC(KP$) > 95 THEN 1370
1420 :
1430 :
1440 REM ** STABILISCE CODICE VISUALIZZAZIONE
1450 :
1460 ADJFAC = (ASC(KP$) > 63)
1470 DSCODE = ASC(KP$) + (ADJFAC * 64)
1480 SA = ROM + (DSCODE * 8)
1490 :
1500 :
1510 REM ** PORTA IN MEMORIA I CARATTERI ROM
1520 :
1530 POKE 56334, PEEK(56334) AND 254
1540 POKE 1, PEEK(1) AND 251
1550 :
1560 :
1570 REM ** MODELLI CARATTERI A MAPPA DI BIT
1580 :
1590 FOR BYTE = 0 TO 7
1600 :    POKE CURSR + BYTE, PEEK (SA + BYTE)
1610 NEXT BYTE
```

```
1620 :
1630 :
1640 REM ** LASCIA ANDARE CARATTERI ROM
1650 :
1660 POKE 1, PEEK(1) OR 4
1670 POKE 56334, PEEK(56334) OR 1
1680 :
1690 :
1700 REM ** REGOLA CURSORE E TORNA IN CICLO
1710 :
1720 CURSR = CURSR + 8
1730 IF CURSR = BASE + 8000 THEN CURSR = BASE
1740 GOTO 1370
1750 :
1760 :
1770 REM ** TORNA A MODO TESTO E FINISCE
1780 :
1790 POKE BSET, PEEK(BSET) AND 223
1800 POKE BLOC, 21
1810 :
1820 PRINT "♥";
1830 END
```

FIG. 6.3. Listato del programma "Testo a mappa di bit".

JOYSTICK

Potete inserire nel vostro Commodore 64 due joystick standard da videogiochi. Vediamo come si possa arrivare all'informazione fornita da un joystick. Poi useremo quell'informazione per far volare uno sprite.

Una leva ha 4 commutatori di direzione, che potete considerare come le direzioni della bussola come mostrato in figura 6.4. In qualsiasi momento possono essere attivati uno o due commutatori. Per esempio, se spingete il joystick a nord viene attivato il commutatore 0. Se lo spingete a sud-ovest vengono attivati i commutatori 1 e 2. Se non lo spingete, non viene attivato alcun commutatore. C'è inoltre un quinto interruttore sul joystick ed è usato come pulsante per sparare.

Ogni commutatore è collegato a un bit in una locazione di input/output del computer.

I cinque commutatori del joystick inserito nel porto di controllo 1 sono collegati ai cinque bit inferiori del registro di input/output nella locazione di memoria 53621. Allo stesso modo i cinque commutatori inseriti nel porto 2 sono collegati ai cinque bit inferiori del registro di input/output nella locazione di memoria 53620. Guardate la figura 6.5. Queste locazioni di input/output sono usate anche dal sistema operativo del computer per esplorare la tastiera. A causa di alcune complicazioni causate dalla tastiera, possono succedere cose strane quando si inserisce

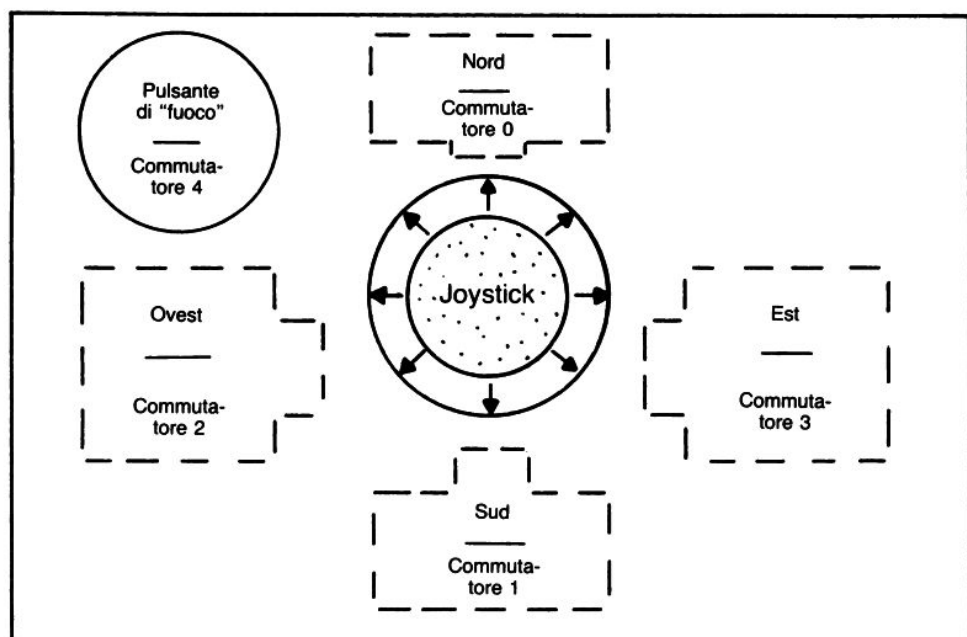



FIG. 6.4. Un Joystick di comando e i suoi cinque commutatori.

un joystick nel porto 1. Così, se state usando solo un joystick, inseritelo nel porto 2.

Potete sapere cosa accade a un joystick leggendo i dati dal corrispondente registro di input/output. Quando un commutatore non è attivato, il bit corrispondente verrà fissato a 1. Quando il commutatore è attivato il bit avrà valore 0. Per esempio, se spostate il joystick a est, essa attiverà il commutatore 3, così il bit 3 del byte di input/output avrà valore 0. Se

Valore del bit →	128	64	32	16	8	4	2	1
Numero del bit →	7	6	5	4	3	2	1	0
	—	—	—	Commutatore 4 — Pulsante di "fuoco"	Commutatore 3 — Est	Commutatore 2 — Ovest	Commutatore 1 — Sud	Commutatore 0 — Nord

Bit 5, 6, 7 usati per altri scopi

FIG. 6.5. Collegamento tra i cinque commutatori e i cinque bit inferiori del registro di input/output nella locazione di memoria 56321 o 56320.

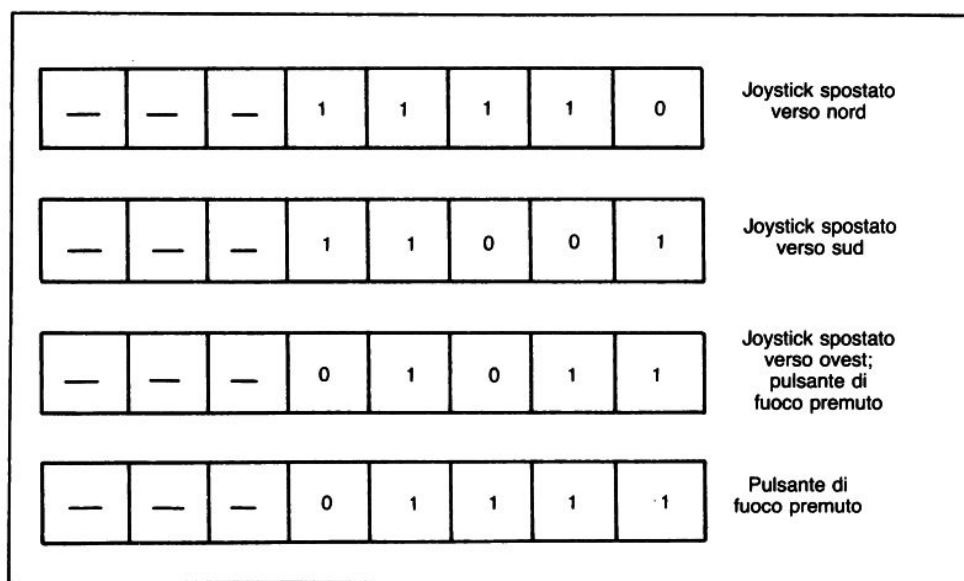



FIG. 6.6. Esempi che mostrano lo stato dei bit nella locazione 56321 quando un joystick viene manipolato in vari modi.

premete il bottone per sparare questo attiva il commutatore 5 e il bit 5 avrà valore 0. La figura 6.6 mostra altri esempi. Usando la funzione AND si possono isolare i bit che si vuole controllare. Basandovi sui risultati, potete ottenere altri valori per la posizione di uno sprite e potrete muoverlo sullo schermo. I programmatori cercano sempre il modo più veloce e intelligente per la lettura del joystick.
Ricordatevi, per quanto complicata appaia la codifica di lettura del joystick, si tratta solo di tradurre i valori dei bit in informazioni di movimento orizzontale e verticale. Nel prossimo paragrafo verrà discusso un programma che usa una di queste tecniche veloci e intelligenti. Ma prima discuteremo dell'identificazione delle collisioni.

OGGETTI CHE SI URTANO SULLO SCHERMO

È utile sapere quando gli oggetti si urtano tra di loro sullo schermo. Con i computer precedenti non era facile. Il Commodore 64 possiede circuiti speciali per l'identificazione delle collisioni.
Le collisioni tra sprite vengono registrate in un registro in VIC + 30 (locazione di memoria 53278). Ogni bit del registro corrisponde a uno sprite. Ogni sprite coinvolto nella collisione riceve nel suo bit il valore 1. Per esempio, se lo sprite 2 urta lo sprite 7, ai bit 2 e 7 di VIC + 30 sarà attribuito il valore 1. I bit manterranno questo valore finché leggerete l'informazione dal registro con un'istruzione PEEK.
Le collisioni fra sprite e dati sono registrate nel registro in VIC + 31 (locazione di memoria 53279). Per dati si intendono parti di caratteri o di immagini a mappa di bit. Ancora, ogni bit del registro corrisponde ad uno sprite e a quel bit è attribuito il valore 1 se il suo sprite si trova coinvolto in una collisione. Per esempio, se lo sprite 5 urta in parte di un carattere, al bit 5 di VIC + 31 sarà assegnato il valore 1. Il bit mantiene quel valore finché il contenuto del registro viene letto.
La figura 2.7 mostra il programma "Urto giocoso"; esso propone esempi di lettura di uno joystick e di identificazione di collisioni tra sprite. Inseritelo, salvatelo e fatelo girare. Appariranno due sprite, come mostra la figura 6.8. Usate un joystick inserito nel porto 2 per muovere la faccia.
Notate cosa accade quando gli sprite si urtano. Digitando il tasto per sparare, il programma terminerà.
Esaminiamo questo programma; le linee 1050-1090 caricano i dati per entrambi gli sprite. Le linee 1380-1540 assegnano i registri necessari del VIC e attivano i due sprite. Ora c'è il segmento principale del programma. La linea 1590 legge il valore della locazione di input/output in 56320. Ricordatevi che questo è il registro che comunica con il joystick inserito nel porto 2.
La linea 1600 usa l'operazione AND per controllare se il tasto per sparare è stato premuto. In caso positivo, il programma esce attraverso la routine di ripulitura che inizia alla linea 1870.
Le linee 1610 e 1620 prendono il valore della locazione 53620 e calcolano il movimento orizzontale e verticale, attraverso una tecnica veloce e alcuni trucchi. L'operazione AND isola i singoli bit che corrispondono ai singoli commutatori nel joystick. La funzione SGN dà valori 0 e 1, dipendenti dal fatto che le espressioni in parentesi assumano il valore 0 o un valore maggiore di 0. La variabile HD avrà assegnato uno di questi valori: − 1, 0, + 1 a seconda del movimento effettuato. Lo stesso vale per VD, la variabile che contiene i valori per il movimento

```
1000 REM *** URTO GIOCOSO ***
1010 :
1020 :
1030 REM ** CARICA I DATI DEGLI SPRITE
1040 :
1050 PRINT "□"
1060 FOR N = 832 TO 958
1070 :   READ SPDTA
1080 :   POKE N, SPDTA
1090 NEXT N
1100 :
1110 DATA   0, 255,   0,   1, 129, 128
1120 DATA   3,   0, 192,   3,   0, 192
1130 DATA   3,   0, 192,   6, 102,  96
1140 DATA  60, 102,  60,  96,   0,   6
1150 DATA 192,   0,   3, 192, 102,   3
1160 DATA 198,  60,  99,  99,   0, 198
1170 DATA 113, 129, 142,  28, 195,  56
1180 DATA  12, 195,  48,  12, 102,  48
1190 DATA   6,  60,  96,   6,   0,  96
1200 DATA   3, 129, 192,   0, 195,   0
1210 DATA   0, 126,   0,   0
1220 :
1230 DATA   0,   0,   0,   0,  16,   0
1240 DATA   0,  56,   0,   0,  84,   0
1250 DATA   0,  16,   0,   2,  16, 128
1260 DATA   1,  17,   0,   0, 146,   0
1270 DATA  16,  84,  16,  32,  56,   8
1280 DATA 127, 255, 252,  32,  56,   8
1290 DATA  16,  84,  16,   0, 146,   0
1300 DATA   1,  17,   0,   2,  16, 128
1310 DATA   0,  16,   0,   0,  84,   0
1320 DATA   0,  56,   0,   0,  16,   0
1330 DATA   0,   0,   0
1340 :
1350 :
1360 REM ** INIZIALIZZA E ATTIVA GLI SPRITE
1370 :
1380 VIC = 53248      :REM CHIP GRAFICO
1390 POKE VIC+33,0    :REM SFONDO NERO
1400 :
1410 POKE 2040,13     :REM SPR0 PUNT DATI
1420 POKE 2041,14     :REM SPR1 PUNT DATI
1430 :
1440 POKE VIC,120     :REM SPR0 ORIZZ
1450 POKE VIC+2,160   :REM SPR1 ORIZZ
1460 POKE VIC+1,138   :REM SPR0 VERT
1470 POKE VIC+3,126   :REM SPR1 VERT
1480 :
1490 POKE VIC+39,3    :REM SPR0 CYAN
1500 POKE VIC+40,7    :REM SPR1 GIALLO
1510 POKE VIC+29,2    :REM SOLO SPR1 E' A
1520 POKE VIC+23,2    :REM DOPPIA DIMENSIONE
1530 :
1540 POKE VIC+21,3    :REM ATTIVA I DUE SPRITE
1550 :
1560 :
1570 REM ** VOLA LO SPRITE 0
1580 :
1590 JR = PEEK (56320) :REM CTRL PORT 2
1600 IF (JR AND 16) = 0 THEN 1870
1610 HD = SGN(JR AND 4) - SGN(JR AND 8)
1620 VD = SGN(JR AND 1) - SGN(JR AND 2)
1630 :
```

```
1640 POKE VIC, PEEK(VIC) + HD
1650 POKE VIC+1, PEEK(VIC+1) + VD
1660 :
1670 :
1680 REM ** IN ASSENZA DI COLLISIONI
1681 REM    TORNA INDIETRO IN CICLO
1690 :
1700 IF PEEK(VIC+30) = 0 THEN 1490
1710 :
1720 :
1730 REM ** COLLISIONE: LO SPRITE 1
1731 REM    DIVENTA BIANCO E LO 0 VIBRA
1732 REM    AD ARCOBALENO
1740 :
1750 POKE VIC+40, 1
1760 :
1770 HUE = PEEK(VIC+39) AND 15
1780 HUE = HUE + 1
1790 IF HUE = 8 THEN HUE = 1
1800 POKE VIC+39, HUE
1810 :
1820 GOTO 1590
1830 :
1840 :
1850 REM ** PULISCE TUTTO E FINISCE
1860 :
1870 POKE VIC+21,0
1880 POKE VIC+29,0
1890 POKE VIC+23,0
1900 :
1910 END
```

FIG. 6.7. Listato del programma "Urto giocoso".

verticale. Questi valori di movimento vengono poi usati per aggiornare la posizione dello sprite 0.

La linea 1700 controlla il registro delle collisioni tra sprite. Se gli sprite non si stanno urtando il programma ritorna a inizializzare i colori

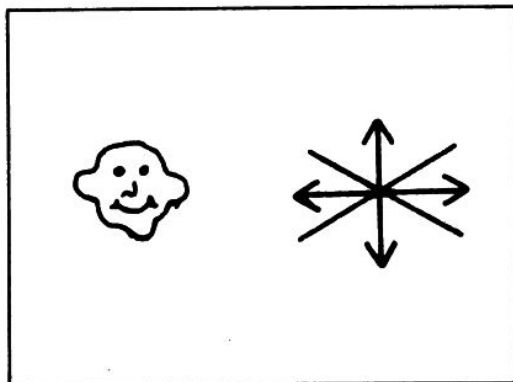



FIG. 6.8. Immagine iniziale mostrata dal programma "Urto giocoso".

originali dello sprite e riguarda il joystick. Se c'è una collisione, le linee 1750-1800 cambiano colore allo sprite prima di ritornare a guardare il joystick.

MODO CARATTERE MULTICOLORE

Nel capitolo 3, abbiamo imparato come creare sprite multicolori. Rinunciando a un po' di risoluzione orizzontale, eravamo in grado di ottenere più colori nel disegno di uno sprite.
C'è anche un modo multicolore per i caratteri. Ancora, si rinuncia a un po' di risoluzione orizzontale per una gamma maggiore di colori. Potete usare questo modo multicolore sia con i caratteri in ROM sia con quelli da voi disegnati. Come per gli sprite multicolore, i caratteri multicolore usano due bit per scegliere un colore. Così ogni riga del carattere sarà costituita da 4 pixel di dimensione doppia. Ricorderete che due bit possono contenere 4 valori possibili: 00, 01, 10, 11.
Questo vi permette di usare 4 colori in un carattere multicolore.
Se si assegna al bit 4 del registro in VIC + 22 (locazione di memoria 53270) il valore 1, si attiva il modo carattere multicolore. Riinizializzando lo stesso bit a 0 lo si disattiva. Per aggiungere un altro controllo (e un'altra complicazione) ogni locazione sullo schermo ha la possibilità di entrare o meno nel modo multicolore. Se la locazione della mappa del colore corrispondente a una locazione dello schermo ha il bit 3 di valore 1, il carattere sarà rappresentato nel modo multicolore. Se il bit 3 della memoria del colore è posto a 0, il carattere manterrà i suoi due colori normali.
Un po' confuso? Proviamo ad affrontare l'argomento in un altro modo. Supponete di aver acceso il modo carattere multicolore, assegnando al bit 4 in VIC + 22 il valore 1. Se inserite nella memoria del colore un numero da 0 a 7, la corrispondente locazione dello schermo rappresenta il suo carattere normalmente. Ma se inserite un numero da 8 a 15 nella locazione della memoria del colore, il carattere verrà rappresentato nel modo multicolore.
Altro dettaglio: se il modo carattere multicolore è attivo e alla locazione della memoria del colore del carattere viene assegnato un numero da 8 a 15, da dove provengono i 4 colori? Se la coppia di bit è 00, i colori provengono dal valore immagazzinato in VIC + 33, il registro del colore dello schermo, chiamato anche registro 0 dello sfondo. Se la coppia di bit è 01, il colore proviene da VIC + 34, registro 1 dello sfondo. Se la coppia di bit è 10 il colore proviene da VIC + 35, registro 2 dello sfondo. Infine se la coppia di bit è 11, il colore proviene dai 3 bit inferiori della locazione di memoria del colore del carattere.

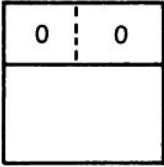
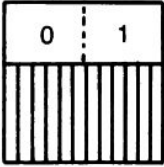
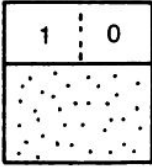
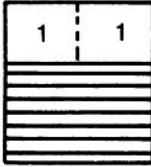
FIG. 6.9. Scheda di codifica che potete usare per disegnare caratteri multicolori.

Se vi fermate un attimo a pensare, vi accorgerete che i caratteri visualizzati nel modo multicolore avranno in comune 3 colori.
Inserendo, con istruzioni POKE, nuovi valori nei 3 registri dello sfondo, si cambierà velocemente l'intero schermo di caratteri multicolori.
Si può usare il modo multicolore con i caratteri già esistenti, ma i risultati non sono molto interessanti. È più divertente disegnarsi i propri caratteri multicolori; la figura 6.9 mostra la scheda di codifica che potete usare allo scopo. La figura 6.10 è un esempio di come può essere usata questa scheda. Vi consiglio di usare quattro colori diversi di riferimento, anche se in questo libro in bianco e nero ho dovuto ricorrere a una distinzione di colore data dall'ombreggiatura.
La figura 6.11 mostra il listato di un programma che mostra i caratteri multicolori.

Valore del bit ▶	128	64	32	16	8	4	2	1	Numero codici
Byte 0	0	1	1	0	1	0	1	1	107
Byte 1	0	1	1	0	1	0	1	1	107
Byte 2	0	1	1	0	1	0	1	1	107
Byte 3	0	1	0	0	0	0	1	1	67
Byte 4	1	1	0	0	0	0	0	1	193
Byte 5	1	1	1	0	1	0	0	1	233
Byte 6	1	1	1	0	1	0	0	1	233
Byte 7	1	1	1	0	1	0	0	1	233

0 0	0 1	1 0	1 1
Colore di sfondo 0 (colore dello schermo)	Colore di sfondo 1	Colore di sfondo 2	Bit inferiori del colore nella memoria del colore

FIG. 6.10. Un esempio che mostra come può essere utilizzata la scheda di codifica di un carattere multicolore.

Inseritelo, salvatelo e fatelo girare. Premendo uno qualsiasi dei tasti **1**, **2**, **3** o **4** potrete cambiare uno dei quattro colori usati nella visualizzazione. Tenendo premuto il tasto avrete un cambiamento continuo del colore. Notate come la figura cambia velocemente quando viene inserito un nuovo valore in uno dei registri dello sfondo. Giocare un po' con questo programma vi insegnerà molto riguardo al modo caratteri multicolori.

Il programma è abbastanza semplice. Il primo segmento carica 2 modelli personalizzati di caratteri e il modello per uno spazio. Poi il video viene pulito; VIC viene regolato in modo che punti al nuovo insieme di caratteri e viene attivato il modo multicolore. Le linee **1360-1440** stampano due linee piene di nuovi caratteri.

Ora viene la parte che esegue il lavoro più pesante.

```
1000 REM *** MULTICOLORE PERSONALIZZATO ***
1010 :
1020 :
1030 REM ** CARICA NUOVI A, B E SPAZIO
1040 :
1050 CBASE = 12288 :REM INIZIO NUOVI CARATTTERI
1060 :
1070 FOR CHAR = 1 TO 2
1080 :    FOR BYTE = 0 TO 7
1090 :       SPOT = CBASE + CHAR*8 + BYTE
1100 :       READ CDTA
1110 :       POKE SPOT, CDTA
1120 :    NEXT BYTE
1130 NEXT CHAR
1140 :
1150 FOR BYTE = 0 TO 7
1160 :    SPOT = CBASE + 32*8 + BYTE
1170 :    POKE SPOT, 0
1180 NEXT BYTE
1190 :
1200 DATA 107, 107, 107,  67
1210 DATA  67, 107, 107, 107
1220 DATA 233, 233, 233, 193
1230 DATA 193, 233, 233, 233
1240 :
1250 :
1260 REM ** PULISCE LO SCHERMO,
1261 REM     INSERISCE NUOVI CARATTERI,
1262 REM     ATTIVA IL MODO MULTICOLORE
1270 :
1280 PRINT "[illegible]";          :REM PULISCE LO SCHERMO
1290 VIC = 53248         :REM CHIP GRAFICO
1300 POKE VIC+24, 29    :REM NUOVI CARATTERI
1310 POKE VIC+22, PEEK(VIC+22) OR 16
1320 :
1330 :
1340 REM ** PREDISPONE VISUALIZZAZIONE
1350 :
1360 PRINT "[illegible]"; :REM GIU' 10
1370 PRINT "[illegible]";      :REM SU   7
1380 PRINT "[illegible]";:REM INIZIA CON COLORE 9
1390 :
1400 FOR N = 1 TO 26
1410 :  PRINT "AB";
1420 :  IF N <> 13 THEN 1440
1430 :      PRINT:PRINT:PRINT "[illegible]";
1440 NEXT N
1450 :
1460 :
1470 REM ** PRESSIONE PULSANTE DI GIOCO
1480 :
1490 COLMAP = 55296
1500 BG = COLMAP + (10 * 40) + 7
1510 POKE 650, 128 :REM TUTTI I TASTI RIPETONO
1520 :
1530 GET KP$
1540 IF KP$ = "" THEN 1530
1550 IF KP$ = " " THEN 1830
1560 :
1570 BKREG = 0
1580 IF KP$ = "1" THEN BKREG = VIC+33
1590 IF KP$ = "2" THEN BKREG = VIC+34
1600 IF KP$ = "3" THEN BKREG = VIC+35
1610 IF KP$ = "4" THEN GOSUB 1720
```

```
1620 IF BKREG = 0 THEN 1530
1630 :
1640 HUE = (PEEK(BKREG) AND 15) + 1
1650 IF HUE = 16 THEN HUE = 0
1660 POKE BKREG, HUE
1670 GOTO 1530
1680 :
1690 :
1700 REM ** SUBROUTINE PER MODIFICARE I COLORI
1701 REM    DELLA MAPPA COLORE DELLE LETTERE
1710 :
1720 HUE = (PEEK(BG) AND 15) + 1
1730 IF HUE > 15 THEN HUE = 8
1740 :
1750 FOR SPOT = BG TO (BG + 106)
1760 :    POKE SPOT, HUE
1770 NEXT SPOT
1780 RETURN
1790 :
1800 :
1810 REM ** PULISCE TUTTO E FINISCE
1820 :
1830 PRINT "□";
1840 POKE VIC+22, PEEK(VIC+22) AND 239
1850 POKE VIC+24, 21
1860 PRINT "■"      :REM TESTO IN BIANCO
1870 POKE VIC+30,0 :REM SU SFONDO NERO
1880 :
1890 END
```

FIG. 6.11. Listato del programma "Multicolore personalizzato".

Il programma riceve l'input di un tasto: se è uno spazio il programma termina, se è 1, 2, 3 o 4 l'appropriata locazione di immagazzinamento del colore viene cambiata. Poi il programma ritorna ad attendere che un altro tasto venga digitato.
Una tecnica che dovete tener presente: quando si legge il colore dalla memoria, un'operazione AND viene usata per eliminare i bit non voluti. Questo succede nelle linee 1640-1720.

MODO CARATTERE A SFONDO ESTESO

C'è un altro modo per visualizzare i caratteri: il modo a sfondo esteso. In questo modo, potete usare uno dei 16 colori per lo sfondo di un carattere. Come sempre, il carattere stesso può assumere qualsiasi dei 16 colori.
Ci sono 4 locazioni di memoria usate per il modo a sfondo esteso, cioè i registri 0-3 dello sfondo, localizzati rispettivamente in VIC + 33, VIC + 34, VIC + 35 e VIC + 36, che comprendono le locazioni di memoria da 53281 a 53284. A ognuna di queste locazioni può essere attribuito uno dei 16 colori.

Come avete visto, per ottenere una visualizzazione più ricca di colore di solito bisogna rinunciare a qualcos'altro. Il modo a sfondo esteso non fa eccezione. Possono essere visualizzati solo 64 caratteri, e non 256. Questo perché i bit 6 e 7 della codifica sono usati per selezionare uno dei registri dello sfondo, il che lascia solo sei bit per codificare il carattere e le leggi dell'aritmetica binaria dicono che sei bit producono 64 valori diversi.
Consideriamo alcuni dettagli pratici. Inserendo 1 nel bit della locazione di memoria 53265, cioè VIC + 17, si attiva il modo colore esteso. Inserendo 0 nella stessa posizione del bit lo si disattiva. Il colore del carattere è immagazzinato nella memoria del colore, come nel normale modo carattere. La codifica del carattere è immagazzinata come sempre nella memoria dello schermo. Tuttavia, vengono usati solo i primi 64 modelli di caratteri. Se i primi due bit della codifica del carattere sono 00, il colore dello sfondo proviene dal registro 0 dello sfondo in VIC + 33. Se i primi 2 bit della codifica sono 01, 10, 11 il colore dello sfondo proviene rispettivamente dai registri 1, 2 o 3 dello sfondo.
Per esempio: se è attivo il modo di colore a sfondo esteso, l'assegnazione del valore 5 a una locazione della memoria dello schermo farà apparire una E sullo schermo. Il colore dello sfondo del carattere verrà dal registro 0 dello sfondo in VIC + 33. Dato che il registro determina il colore dello sfondo per tutto lo schermo, la E apparirà del tutto normale.
Assegnando 69 in una locazione della memoria dello schermo apparirà una E sullo schermo, ma all'area 8 * 8 del carattere verrà assegnato un colore di sfondo basato sul contenuto di VIC + 34. Allo stesso modo, l'inserimento di 133 produrrà una E con il colore locale di sfondo basato sui contenuti di VIC + 35.
L'inserimento di 197 nella memoria dello schermo produrrà una E e un colore di sfondo basato sul contenuto di VIC + 36.
La figura 6.12 mostra il listato del programma "Sfondo esteso", il che dà una dimostrazione di questo modo. Inseritelo, salvatelo e fatelo girare. Ogni colonna di lineette ha in comune lo stesso registro di sfondo. Premendo uno dei tasti da 1 a 4 cambierà il contenuto di uno dei registri di sfondo. Premendo 5 cambierà il colore del carattere. Ancora una volta, se volete veramente capire un nuovo modo, dovete passare del tempo a modificare il programma.
Una breve spiegazione del programma "Sfondo esteso". Le linee 1050-1070 puliscono lo schermo e attivano il modo sfondo esteso. Le linee 1120-1250 costruiscono 4 colonne di lineette (codice di visualizzazione: 45). Comunque ogni colonna differisce nei bit 6 e 7, cosicché le colonne di lineette faranno riferimento a registri diversi, per quanto riguarda i colori di sfondo.

```
1000 REM *** SFONDO ESTESO ***
1010 :
1020 :
1030 REM ** ATTIVA MODO SFONDO ESTESO
1040 :
1050 PRINT "♥";        :REM PULISCE SCHERMO
1060 VIC = 53248       :REM CHIP GRAFICO
1070 POKE VIC+17, PEEK(VIC+17) OR 64
1080 :
1090 :
1100 REM ** PREDISPONE VISUALIZZAZIONE
1110 :
1120 SCREEN = 1024
1130 COLMAP = 55296
1140 SS = SCREEN + (10 * 40) + 16
1150 CS = COLMAP + (10 * 40) + 16
1160 :
1170 HUE = PEEK(CS) + 1
1180 IF HUE = 16 THEN HUE = 0
1190 :
1200 FOR RW = 0 TO 3
1210 :FOR N = 0 TO 3
1220 : POKE SS + RW*40 + N*2, 45 + 64*N
1230 : POKE CS + RW*40 + N*2, HUE
1240 : NEXT N
1250 NEXT RW
1260 :
1270 :
1280 REM ** PRESSIONE PULSANTE DI GIOCO
1290 :
1300 POKE 650, 128 :REM TUTTI I TASTI RIPETONO
1310 :
1320 GET KP$
1330 IF KP$ = "" THEN 1320
1340 IF KP$ = " " THEN 1520
1350 :
1360 BKREG = 0
1370 IF KP$ = "1" THEN BKREG = VIC+33
1380 IF KP$ = "2" THEN BKREG = VIC+34
1390 IF KP$ = "3" THEN BKREG = VIC+35
1400 IF KP$ = "4" THEN BKREG = VIC+36
1410 IF KP$ = "5" THEN 1170
1420 IF BKREG = 0 THEN 1320
1430 :
1440 HUE = (PEEK(BKREG) AND 15) + 1
1450 IF HUE = 16 THEN HUE = 0
1460 POKE BKREG, HUE
1470 GOTO 1320
1480 :
1490 :
1500 REM ** PULISCE E FINISCE
1510 :
1520 PRINT "♥";
1530 POKE VIC+17, PEEK(VIC+17) AND 191
1540 PRINT "■"      :REM TESTO IN BIANCO
1550 POKE VIC+33,0 :REM SU SFONDO NERO
1560 :
1570 END
```

FIG. 6.12. Listato del programma "Sfondo esteso".

La sezione successiva è un altro grande ciclo di interrogazione della tastiera. Premendo la barra spazio il programma termina; digitando i numeri da 1 a 5 i colori cambiano come abbiamo già visto e qualsiasi altro tasto è ignorato. Infine, l'ultimo modulo pulisce tutto: disattiva il modo sfondo esteso, pulisce lo schermo e attribuisce al carattere il colore bianco.

MODO MAPPA DI BIT MULTICOLORE

C'è un'ultima opzione di visualizzazione del Commodore 64: il modo mappa di bit multicolore. Come avrete indovinato, questo modo di funzionamento vi dà la possibilità di usare 4 colori in un blocco di 8 × 8 della visualizzazione a mappa di bit. Probabilmente avete anche indovinato il costo: risoluzione orizzontale dimezzata. Come viene attivato questo modo? Primo, inserite un 1 nel bit 5 in VIC + 17 per attivare il modo mappa di bit. Poi dite a VIC dove è localizzata la mappa di bit da 8K fissando il bit 3 di VIC + 24. Nella maggior parte dei casi, a quel bit attribuirete valore 1. Fin qui, sono esattamente i passi che avete usato per inizializzare la normale mappa di bit. Infine, inizializzate a 1 il bit 4 in VIC + 22, il quale attiva il modo multicolore.
La corrispondenza tra byte nella mappa di bit e i punti sullo schermo è la stessa della normale mappa di bit. Tuttavia, due bit vengono usati per scegliere un colore per un pixel di larghezza doppia. Come ormai saprete, 2 bit possono codificare 4 valori. In dipendenza del valore di una coppia di bit, l'informazione del colore per una data area 8 × 8 può provenire da una fra 4 locazioni.
Se la coppia di bit è 00, il colore proviene dal registro 0 dello sfondo in VIC + 33, che è lo stesso colore dello sfondo dello schermo. Se la coppia di bit è 01, il colore proviene dal nibble superiore della corrispondente locazione della memoria dello schermo. Se la coppia di bit è 10 il colore proviene dal nibble inferiore dello stesso byte della memoria dello schermo. Infine se la coppia di bit è 11 il colore proviene dalla corrispondente locazione della memoria del colore.
Per ritornare da questo modo a una visualizzazione standard in modo testo dovete solo capovolgere i passi di inizializzazione. Ciò significa inserire uno 0 nel bit 5 di VIC + 17, inserire uno 0 nel bit 4 di VIC + 22 e rieinizializzare VIC + 24 attribuendogli il valore 21.

```
1000 REM *** TESTO A MAPPA DI BIT COLORE ***
1010 :
1020 :
1030 REM ** INIZIALIZZAZIONI VARIE
1040 :
1050 PRINT "J";        :REM PULISCE SCHERMO
1060 POKE 650, 128 :REM TASTI RIPETONO
1070 ROM = 53248    :REM ROM CARATTERI
1080 BASE = 8192    :REM BASE MAPPA BIT
1090 CUR6R = BASE   :REM CURSORE MAPPA BIT
1100 VIC = 53248    :REM CHIP GRAFICO
1110 BLOC = VIC+24 :REM LOCALIZZA MAPPA
1120 BSET = VIC+17 :REM PREDISPONE MMB
1130 :
1140 :
1150 REM ** ATTIVA IL MODO MAPPA DI BIT
1160 :
1170 POKE BLOC, PEEK(BLOC) OR 8
1180 POKE BSET, PEEK(BSET) OR 32
1190 :
1200 :
1210 REM ** FISSA A CASO I COLORI DELLA MAPPA DI BIT
1220 :
1230 FOR SL = 1024 TO 2023
1240 :    POKE SL, (INT(RND(1) * 15) + 1) * 16
1250 NEXT SL
1260 :
1270 :
1280 REM ** RIEMPIE A CASO LA MAPPA
1290 :
1300 FOR BMLOC = BASE TO BASE + 7999
1310 :    POKE BMLOC, INT(RND(1) * 256)
1320 NEXT BMLOC
1330 :
1340 :
1350 REM ** PRENDE UNA LETTERA, UN NUMERO,
1351 REM     UN SEGNO DI INTERPUNZIONE
1360 :
1370 GET KP$
1380 IF KP$ = "" THEN 1370
1390 IF KP$ = " " THEN 1790
1400 IF ASC(KP$) < 32 THEN 1370
1410 IF ASC(KP$) > 95 THEN 1370
1420 :
1430 :
1440 REM ** CALCOLA IL CODICE DI VISUALIZZAZIONE
1450 :
1460 ADJFAC = (ASC(KP$) > 63)
1470 DSCODE = ASC(KP$) + (ADJFAC * 64)
1480 SA = ROM + (DSCODE * 8)
1490 :
1500 :
1510 REM ** PORTA IN MEMORIA LA ROM DEI CARATTERI
1520 :
1530 POKE 56334, PEEK(56334) AND 254
1540 POKE 1, PEEK(1) AND 251
1550 :
1560 :
1570 REM ** MODELLI DI CARATTERI A MAPPA DI BIT
1580 :
1590 FOR BYTE = 0 TO 7
1600 :    POKE CURSR + BYTE,PEEK (SA + 7 - BYTE)
1610 NEXT BYTE
1620 :
```

```
1630 :
1640 REM ** LASCIA ANDARE CARATTERI ROM
1650 :
1660 POKE 1, PEEK(1) OR 4
1670 POKE 56334, PEEK(56334) OR 1
1680 :
1690 :
1700 REM ** REGOLA IL CURSORE E TORNA IN CICLO
1710 :
1720 CURSR = CURSR + 8
1730 IF CURSR = BASE + 8000 THEN CURSR = BASE
1740 GOTO 1370
1750 :
1760 :
1770 REM ** TORNA AL MODO TESTO E FINISCE
1780 :
1790 POKE BSET, PEEK(BSET) AND 223
1800 POKE BLOC, 21
1810 :
1820 PRINT "";
1830 END
```

FIG. 6.13. Mappa di bit multicolore.

CHE COSA ABBIAMO IMPARATO

Questo capitolo è stato ricco di argomenti; ho voluto infatti raccogliere un numero di temi diversi prima di introdurre il prossimo grande argomento: i suoni. Ecco una panoramica degli argomenti che abbiamo trattato:

- Come si muovono gli sprite davanti e dietro altre immagini, attribuendo le priorità fra sprite e sfondo
- Come si inseriscono i caratteri su un'immagine a mappa di bit trasferendo 8 byte dalla memoria dei caratteri
- Come si legge un joystick guardando i cinque bit inferiori delle locazioni di memoria 56320 e 56321
- Come si usa l'informazione del joystick per muovere sprite
- Come si identificano le collisioni tra sprite e tra sprite e altre immagini
- Come si visualizzano caratteri in modo multicolore, dove possono essere usati per ogni carattere 4 colori
- Come si visualizzano i caratteri in modo sfondo esteso, dove tutti i 16 colori sono utilizzabili per sfondo locale
- Come si attiva il modo mappa di bit multicolore, dove possono essere usati 4 colori per ogni blocco 8 × 8 della mappa di bit, sebbene la risoluzione orizzontale venga dimezzata

ESERCIZI

Test

1. Quali sprite si muoveranno dietro immagini di sfondo se nel registro VIC + 27 viene inserito il numero 85?
2. Date un esempio di una situazione in cui torna utile la strana disposizione dei byte nella mappa di bit.
3. In quale direzione viene orientato il joystick se il registro di input/output in 56321 contiene il valore 26?
4. Se il registro di collisione tra sprite contiene il valore 170, quali sprite si sono urtati?
5. Inizializzando a .... il bit .... del registro VIC + 22 si ha l'attivazione del modo carattere multicolore.
6. Nel modo sfondo esteso i bit .... e .... del codice di visualizzazione di un carattere selezionano uno dei 4 registri dello sfondo.
7. Quali 3 bit devono essere considerati per attivare il modo mappa di bit multicolore?

Esercizi di programmazione

Questi esercizi dovrebbero essere di facile programmazione. Soluzioni possibili sono date più avanti. L'esercizio 2 proverà la vostra abilità nell'esaminare criticamente i programmi stampati nel capitolo.

1. Cambiate il programma "Sopra e sotto" in modo che lo sprite si muova in un'orbita verticale (anziché orizzontale).
2. Studiate il programma illustrato in figura 6.13. Questo programma è una revisione del programma "Testo in mappa di bit" il quale stampa i caratteri del testo a colori, ribaltati, su uno sfondo nero. Identificare le linee che determinano questi cambiamenti.
3. Cambiate il programma "Urto giocoso" in modo che il joystick operi al contrario. Muovendo il joystick ad ovest, cioè, lo sprite si sposterà verso est, muovendolo a nord lo sprite andrà a sud e così via.

Risposte al test

1. Gli sprite 0, 2, 4 e 6
2. Quando si vogliono inserire caratteri in una visualizzazione a mappa di bit

3. Nord-ovest
4. Gli sprite 1, 3, 5 e 7
5. 4; 1
6. 6; 7
7. Bit 5 di VIC + 17 (53265); bit 3 di VIC + 24 (53272); bit 4 di VIC + 22 (53270)

Soluzioni possibili agli esercizi di programmazione

1. Caricate il programma "Sopra e sotto" e inserite queste linee:

```
1000 REM *** SOPRA E SOTTO VERTICALE ***
1260 POKE VIC, 172    :REM SPR0 POS ORIZZONTALE
1270 POKE VIC+1, 68   :REM SPR0 POS VERTICALE
1370 :    POKE VIC+1, PEEK(VIC+1) + DR
```

2. Le linee seguenti sono quelle che hanno causato i cambiamenti appropriati:

```
1000 REM *** TESTO A MAPPA DI BIT COLORE ***
1240 :    POKE SL, (INT(RND) * 15) + 1) * 16
1600 :    POKE CURSR + BYTE, PEEK (SA + 7 - BYTE)
```

3. Caricate il programma "Urto giocoso" e inserite queste linee:

```
1000 REM *** URTO SELVAGGIO ***
1610 HD = SGN(JR AND 8) - SGN(JR AND 4)
1620 VD = SGN(JR AND 2) - SGN(JR AND 1)
```

CAPITOLO SETTE

Produzione di suoni

Fino ad ora abbiamo trattato figure mute; in questo capitolo cominceremo a fare un po' di rumore. Tanto per cominciare, rapidamente, vedremo qualche nozione fondamentale sulla natura dei suoni. Impareremo tutto sulla frequenza, l'ampiezza e le forme d'onda. Considereremo il SID, l'efficace integrato del suono del Commodore che è inserito nel vostro computer. Impareremo ad usare i registri del SID. Parleremo di musica e termineremo con una melodia familiare.

ALCUNI ASPETTI DEL SUONO

Tutto ciò che vibra crea suono. Il classico esempio per coloro che iniziano è un diapason. Se ne avete uno, percuotetelo. Ascoltate per un momento e poi toccatelo. Sentite le vibrazioni? Se non avete un diapason a portata di mano, potete eseguire il seguente esperimento: prendete 2 fili di seta o 2 corde, ognuna lunga 65 cm circa.
Poi prendere la reticella da un forno. Attaccate un'estremità di un filo a un angolo della reticella, poi attaccate il secondo filo a un altro angolo. Avvolgete l'estremità libera di un filo intorno al vostro indice sinistro, poi avvolgete l'altra estremità dell'altro filo intorno al vostro indice destro. Mettete le vostre dita nelle orecchie. Urtate la reticella contro qualcosa e ascoltate. Guardare la figura 7.1.

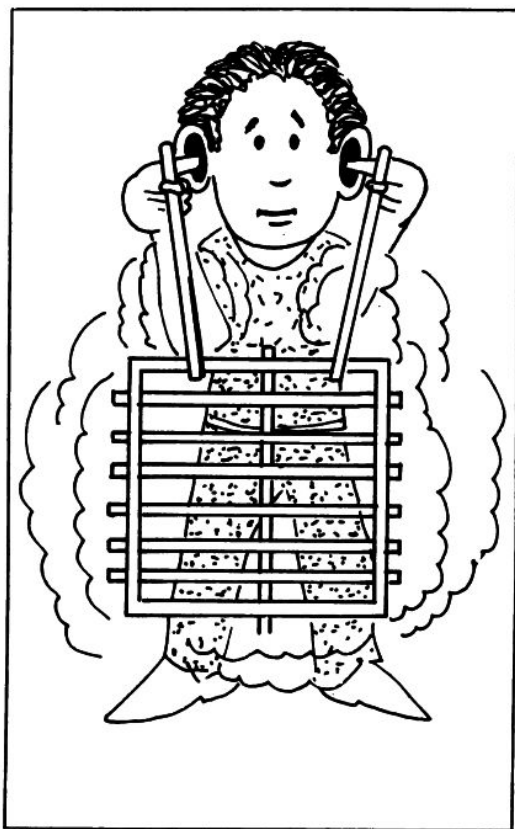



FIG. 7.1. Forse volete provare privatamente a fare questo esperimento di acustica...

Onde

Una vibrazione completa determina un'onda. Gli oggetti che vibrano determinano molte onde. Queste onde si propagano, in particolare nei solidi. Si propagano anche nell'aria. Quando le onde sonore penetrano nel nostro orecchio, si urtano con piccoli peli sensibili, mettendoli in vibrazione. I peli sono collegati ai nervi, che mandano messaggi al cervello. In questo modo sentiamo i suoni.

Frequenza, ovvero altezza

Vi sono molti modi per descrivere le onde. Un modo è quello di contare quante onde ovvero quanti "cicli" si formano in un certo periodo di tempo. Si parla allora della "frequenza" delle onde. Per esempio, se foste nell'oceano, contereste il numero di onde che si formano in un minuto. Se ci fossero 12 onde, potreste dire che la frequenza è di 12 cicli al minuto. Le onde sonore hanno una frequenza superiore, che si misura in cicli al secondo o hertz (Hz). Quella che chiamiamo altezza di un suono dipende dalla frequenza. I suoni bassi, profondi, hanno una bassa frequenza, quelli invece acuti hanno frequenza elevata. Siamo in grado di udire i suoni con frequenza compresa fra 15 e 20000 hertz. Un pianoforte può creare suoni con frequenza fra i 33 e i 4186 hertz. Il vostro computer può creare suoni con frequenza fra 0.6 e 3995 hertz. Potete disegnare le onde dei suoni. La figura 7.2 mostra un modello di onde create da un diapason. Le onde hanno diverse frequenze.

Ampiezza

Si può anche misurare la dimensione di un'onda. Questa è chiamata "ampiezza". Nel caso delle onde sonore, l'ampiezza si traduce in intensità, o volume (la caratteristica che rende un suono più o meno "forte"). Quanto maggiore è l'ampiezza delle onde, tanto più intensi sono i suoni. Frequenza e ampiezza sono caratteristiche indipendenti. Due suoni possono avere la stessa altezza e intensità diverse, oppure la stessa intensità ma altezze diverse.

Forme d'onda

Le onde possono avere diverse forme. Le onde mostrate nelle figure 7.2 e 7.3, prodotte dal diapason, sono chiamate onde sinusoidali: hanno una forma semplice e regolare. La figura 7.4 mostra altre quattro forme d'onda: un'onda triangolare, una a dente di sega, una rettangolare e una complessa. Onde diverse creano suoni con timbri diversi. Il suono del clarinetto che produce un do centrale a un dato volume è diverso dal suono di un pianoforte che produce la stessa nota allo stesso volume, perché la forma delle onde del clarinetto è diversa da quella delle onde del pianoforte. Le forme d'onda sono indipendenti dalla frequenza e dall'ampiezza. Se riguardate la figura 7.4, noterete che ho disegnato le quattro onde con la stessa frequenza e la stessa ampiezza.

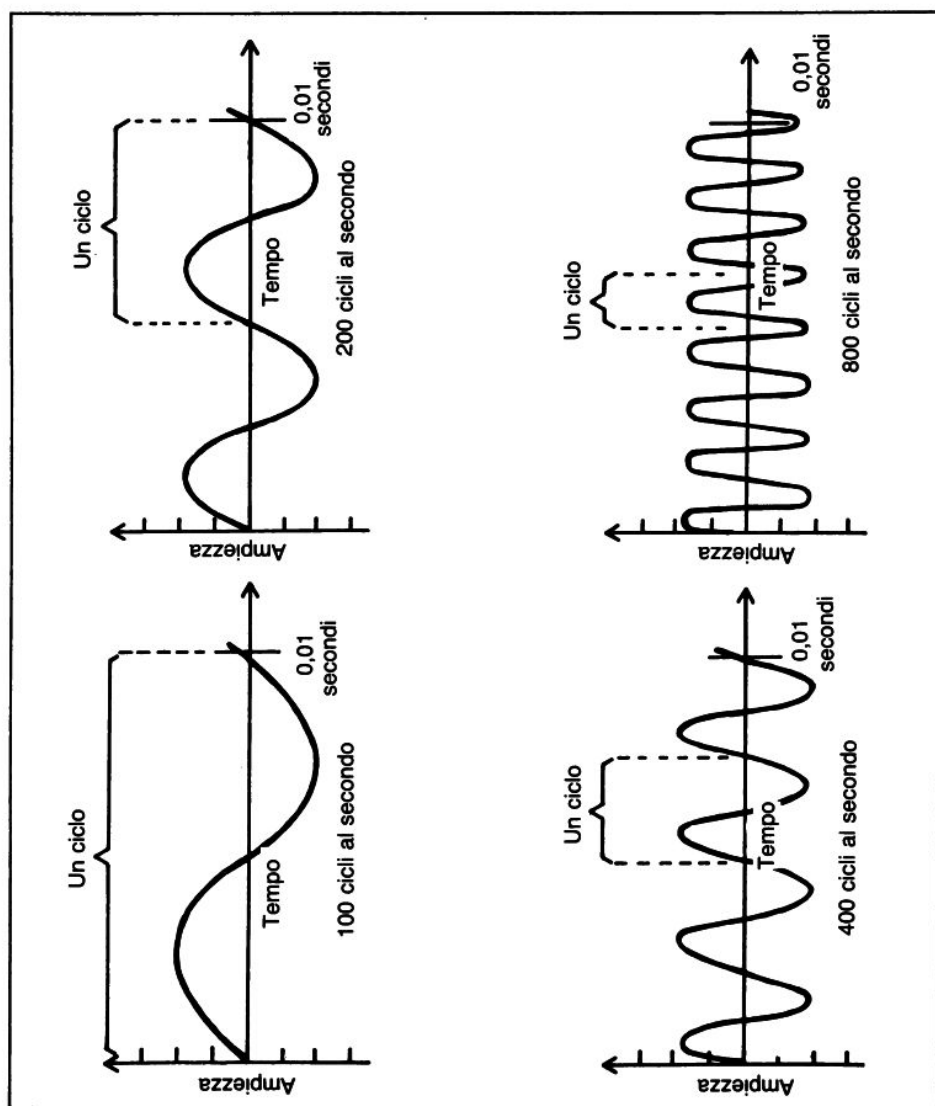



FIG. 7.2. Grafici di onde prodotte da un diapason a diverse frequenze; le onde hanno la stessa ampiezza.

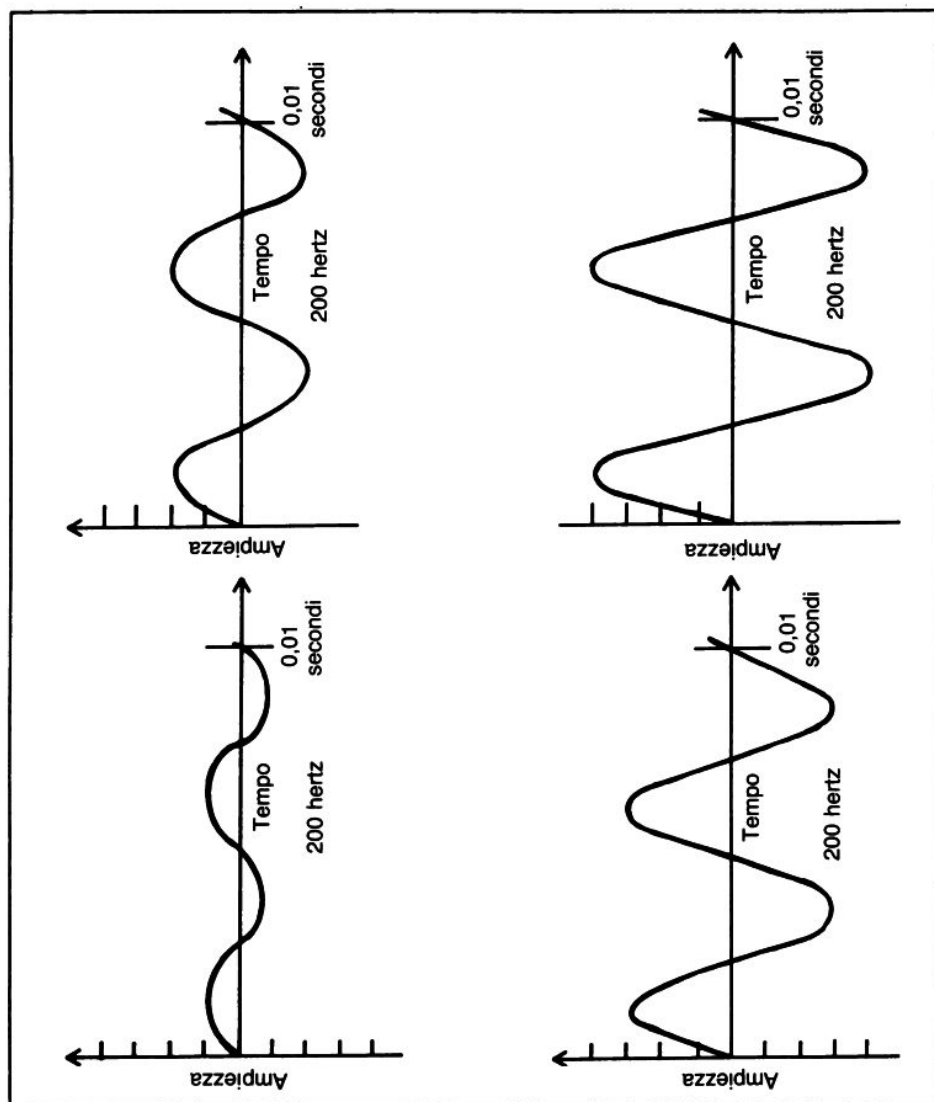



FIG. 7.3. Grafici di onde prodotte da un diapason che hanno la stessa frequenza, ma diversa ampiezza.

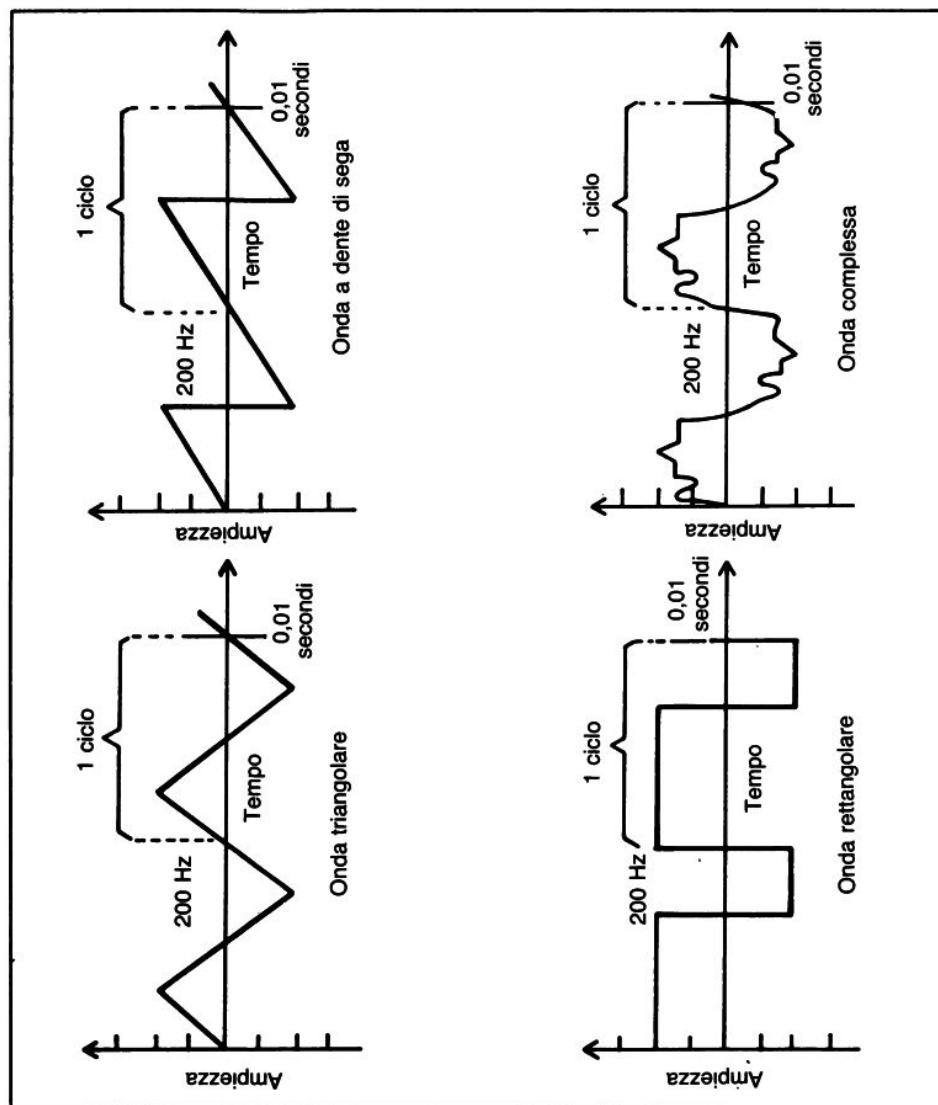



FIG. 7.4. Ancora 4 grafici che mostrano 4 forme d'onda diverse: triangolari, a dente di sega, rettangolari e complesse.

UN BREVE INTERVALLO

Il vostro Commodore 64 può creare molti suoni diversi. Ma questa versatilità ha un prezzo: la complessità. Impiegheremo un po' di tempo prima di imparare a regolare tutti i controlli dei suoni.
Nel frattempo, solo per provare che il Commodore 64 può produrre suoni, fate girare il piccolo programma in figura 7.5. Quando siete stanchi di sentire questa melodia, che si ripete, premete un tasto qualsiasi per farlo terminare.
Resisterò alla tentazione di spiegarvi il programma, poiché quando avrete imparato abbastanza sul SID sarete in grado di capirlo da soli.

SID, IL DISPOSITIVO DI INTERFACCIA DEL SUONO

Avete conosciuto VIC-II, l'integrato della grafica del Commodore 64. Bene, preparatevi a incontrare il SID, integrato per la generazione dei suoni del Commodore 64.
SID sta per *Sound Interface Device*, dispositivo di interfaccia del suono. La Commodore ha inserito un sofisticato sintetizzatore musicale su un singolo circuito integrato. Consideriamo alcune caratteristiche di SID: ha 3 generatori separati di suoni, che sono chiamati *voci*. Si possono usare una, due o tutte e tre le voci per creare i suoni.
Vi sono molti modi per controllare ogni voce. Il generatore di ogni voce è un oscillatore. Regolando i registri opportuni si può fare in modo che l'oscillatore produca onde sonore a qualsiasi frequenza fra 0 e 3995 hertz, che è circa l'estensione del pianoforte.
Ogni voce ha anche un generatore di forme d'onda. Si può scegliere una

```
1000 REM *** SIRENA MINIMA ***
1010 POKE 54296,15     :REM VOLUME SU HI
1020 POKE 54278,240    :REM REGOLA SUSTAIN
1030 POKE 54276,33     :REM NOTA ATTIVA
1040 FOR N = 1 TO 100  :REM SIRENA
1050 :    POKE 54273, 15 + ABS (50 - N)
1060 NEXT N
1070 GET KP$           :REM ANCORA ?
1080 IF KP$ = "" THEN 1040
1090 POKE 54276,0      :REM NOTA SPENTA
1100 POKE 54296,0      :REM VOLUME A ZERO
```

FIG. 7.5. Listato del programma "Sirena minima".

delle quattro forme d'onda per una voce: triangolare, a dente di sega, a impulso o irregolare (rumore). Le onde triangolari e a dente di sega sono mostrate in figura 7.4. A impulsi è solo un altro modo per chiamare la rettangolare: anche queste onde sono mostrate in figura 7.4.
L'onda di rumore è un segnale casuale che assomiglia a quello che riceve il televisore quando tutte le stazioni finiscono di trasmettere. È molto importante per gli effetti sonori. È, propriamente, il "rumore bianco".
Infine ogni voce ha il proprio generatore di inviluppo e un modulatore di ampiezza. Questi meccanismi vi permettono di controllare il volume di ciascuna voce in modo molto preciso. Se pizzicate una nota su una chitarra, noterete che il volume del suono cambia, nel corso della sua durata. Il generatore dell'inviluppo e il modulatore d'ampiezza vi permettono di controllare l'andamento di una voce del SID allo stesso modo.
Ogni voce del SID usa 7 registri. SID contiene un totale di 29 registri. 8 registri permettono di controllare il volume globale di tutte le voci, miscelare e sincronizzare le voci in modi divertenti, filtrare certe frequenze, includere i suoni di fonti esterne, leggere il paddle dei giochi e controllare l'uscita della voce 3. Consideriamo ora più dettagliatamente la regolazione di alcuni registri di SID.

SCHEMA DEI REGISTRI DI SID

I 29 registri del SID occupano le locazioni di memoria **54272-54300**. Come ho fatto per VIC, mi riferirò di solito a registri specifici attraverso la loro posizione relativa nell'insieme dei registri. Ad esempio, ci riferiremo al registro **54278** come **SID + 6**. L'appendice L mostra lo schema completo di un registro del SID. I primi 7 registri controllano la voce 1, i 7 successivi la voce 2, gli altri 7 la voce 3.
I successivi quattro registri controllano i filtri e il volume generale. Gli ultimi quattro registri controllano diverse funzioni.
I tre gruppi di registri che controllano le voci (d'ora in poi li chiamerò "gruppi di registri di voce") vengono inizializzati circa allo stesso modo. Andando avanti vedremo le eccezioni.

REGOLAZIONE DI FREQUENZA

I primi 2 registri di un gruppo di registri di voce controllano la frequenza di quella voce. Ciò significa che i registri in **SID** e in **SID + 1** regolano la

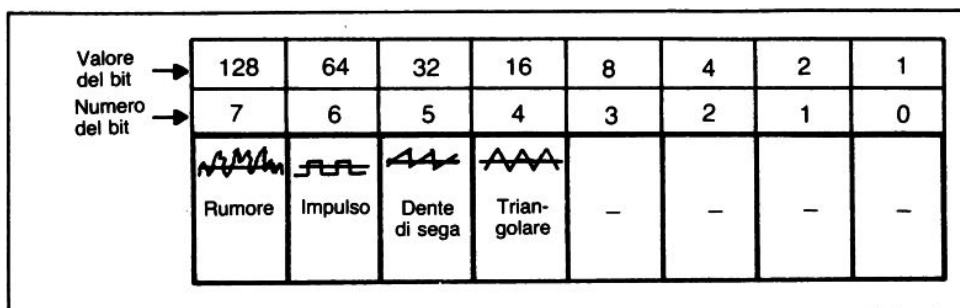



FIG. 7.6. I bit 4, 5, 7 del quinto registro di una voce vengono usati per selezionare la forma d'onda.

frequenza per la voce 1, SID + 7 e SID + 8 la regolano per la voce 2, SID + 14 e SID + 15 per la voce 3. Due registri di 8 bit danno un totale di 16 bit. I valori fra 0 e 65535 possono essere rappresentati con 16 bit. Così ci sono 65536 possibili frequenze per ogni voce.
Come determinare i valori da assegnare ai due registri di frequenza? Prima fate una piccola conversione. Dividete la frequenza in hertz per un fattore particolare e poi arrotondate il risultato al numero intero più vicino. Avrete così il valore della frequenza per il SID. Il fattore si basa sulla velocità dell'orologio del computer ed è 0.060952, più o meno un milionesimo. Per esempio, se volete una frequenza di 440 hertz, arrotondando 440 diviso per 0.0609592 con l'intero più vicino, si ottiene un valore di frequenza di 7218.
Ora dovete convertire il valore della frequenza in due valori da assegnare con le istruzioni POKE ai registri di frequenza. Data la complessità delle basi 2, 10, 16 dividete il valore per 256. La parte intera sarà assegnata al secondo registro di frequenza (SID + 1, SID + 8 o SID + 15), il byte "alto" del valore della frequenza. Il resto della divisione verrà inserito nel primo registro di frequenza (SID, SID + 7 o SID + 14), il byte "basso" del valore di frequenza. Continuiamo con l'esempio del suono da 440 hertz. Avete ottenuto una frequenza di 7218. Dividetela per 256. La parte intera del risultato è 28; il resto è 50. Se volete inizializzare la voce 1 in modo che produca un suono di 440 hertz, dovete assegnare a SID + 1 il valore 28 e a SID il valore 50.

SELEZIONE DELLA FORMA D'ONDA

Il nibble superiore (bit 4, 5, 6, 7) del quinto registro in ogni gruppo, seleziona la forma d'onda per quella particolare voce. SID + 4 è il

registro usato per la voce 1, mentre **SID + 11** e **SID + 18** eseguono il lavoro rispettivamente per le voci 2 e 3.
Inizializzando a 1 uno di questi bit si seleziona la forma dell'onda associata a quel bit. Il bit 4 seleziona un'onda triangolare; il bit 5 seleziona un'onda a dente di sega; un bit 6 seleziona un'onda rettangolare (a impulso); un bit 7 seleziona il rumore bianco.
Guardate la figura 7.6. Se scegliete la forma rettangolare dovete precisare un'altra caratteristica: la larghezza dell'impulso. Vediamo come.

REGOLAZIONE DELLA LARGHEZZA DELL'IMPULSO

Nella forma d'onda rettangolare, o a impulso, l'ampiezza è alta o bassa, senza valori intermedi. La percentuale del ciclo nell'arco della quale l'onda ha ampiezza elevata, è la larghezza dell'impulso.
La figura 7.7 mostra quattro forme d'onda a impulso con larghezza diversa.
I registri 3 e 4 di un gruppo di registri di voce controllano la larghezza dell'impulso (se viene selezionata la forma d'onda rettangolare).
Quali valori assegnamo a questi 2 registri per una data larghezza d'impulso? Prendete il valore della larghezza (espressa come percentuale) e moltiplicatela per 40.95. Arrotondate quel numero e otterrete il valore da inserire nel SID.
Ora dividete il valore della larghezza d'impulso per 256. Assegnate la parte intera del risultato al quarto registro del gruppo della voce. Mettete il resto nel terzo registro del gruppo.
Ecco un esempio: volete una larghezza d'impulso del 75% per la terza voce.
75 × 40.95 = 3071.25, che arrotondato è 3071. 3071 : 256 = 11 con un resto di 255. Allora inserite il valore 11 in **SID + 17** e il valore 255 in **SID + 16.**

ADSR: IL GENERATORE D'INVILUPPO

Abbiamo detto che ogni voce ha un generatore di inviluppo e un modulatore di ampiezza. Questi meccanismi danno un controllo preciso dell'andamento di un suono, in termini di intensità. Il segreto di questo controllo è l'inviluppo ADSR.
ADSR sta per *Attack decay sustain release* (attacco, decadimento, sostegno, rilascio). Queste parole definiscono 4 fasi dell'esistenza del suono.

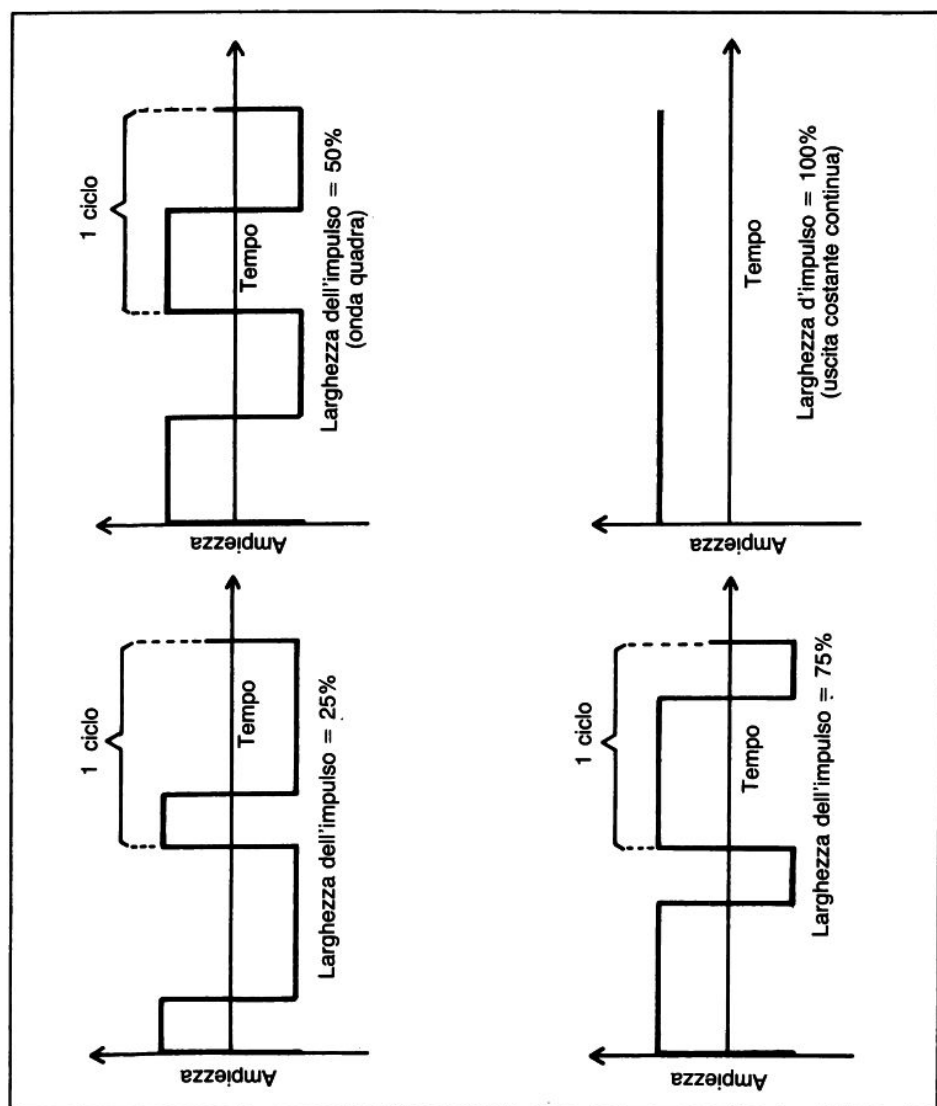



FIG. 7.7. Quattro onde rettangolari con larghezza d'impulso diversa.

Durante il primo stadio il volume passa da 0 al valore massimo.
La velocità d'attacco determina il tempo impiegato per questo passaggio. Durante il secondo stadio, il volume passa dal valore massimo al valore minimo. La velocità di decadimento determina il tempo necessario per questo passaggio. Il livello a cui il volume scende è detto livello di sostegno. Esso può essere espresso come percentuale del volume massimo. Durante il terzo stadio il volume rimane a questo livello. Infine, la nota si estingue. La velocità con cui cade dal livello di sostegno al volume 0 è chiamata velocità di rilascio. Guardate la figura 7.8. Essa mostra i 4 stadi della vita tipica di una nota.
Il sesto e il settimo registro di ogni gruppo di voce definiscono l'inviluppo ADSR. Quando una voce è attivata, i valori nei registri di ADSR controllano il generatore di inviluppo della voce. A sua volta il generatore dell'inviluppo controlla il modulatore di ampiezza.
Il modulatore di ampiezza prende le onde che provengono dall'oscillatore e dal generatore di forma d'onda e ne regola l'ampiezza.

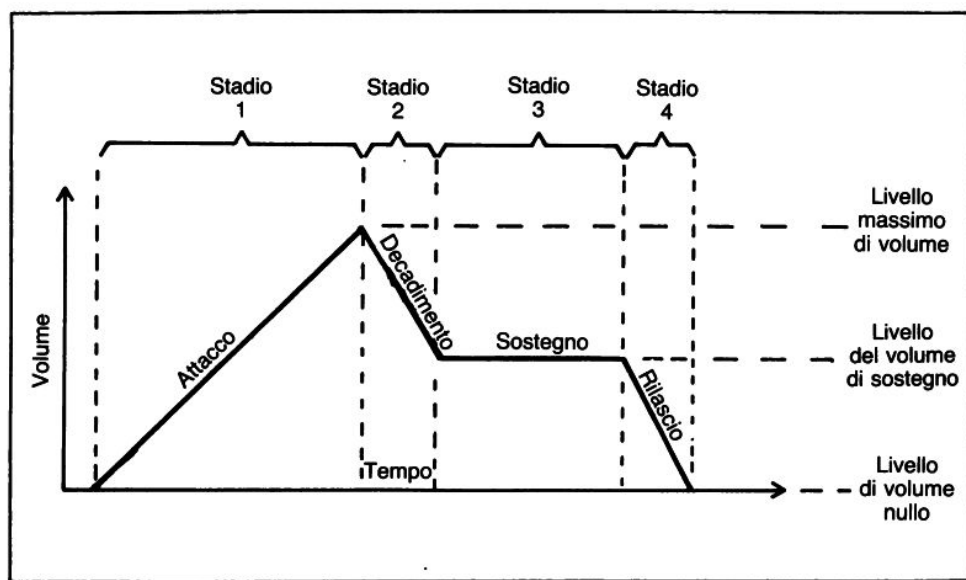



FIG. 7.8. I 4 stadi dell'andamento di una tipica nota, con le variazioni di intensità che determinano l'inviluppo ADSR.

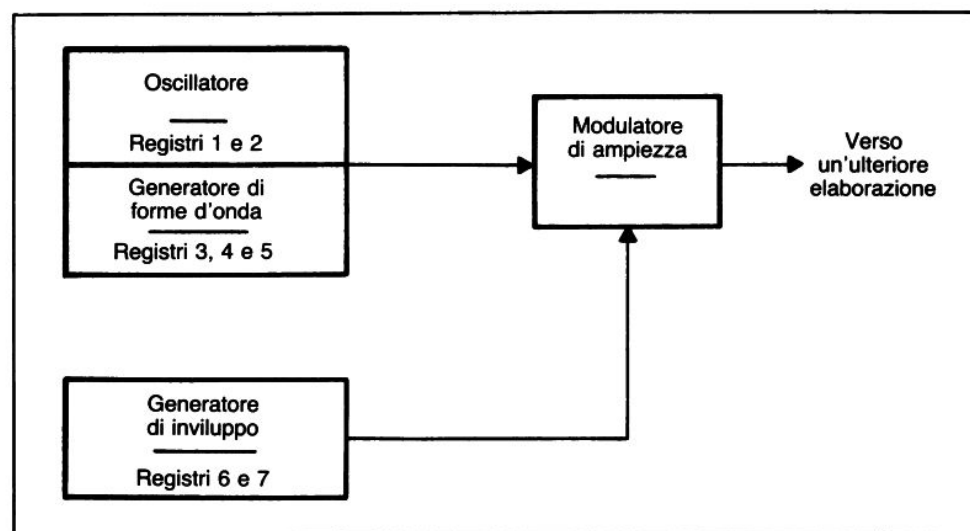



FIG. 7.9. Le informazioni provenienti da un oscillatore, dal generatore della forma d'onda e dal generatore d'inviluppo arrivano al modulatore di ampiezza; il segnale che viene emesso poi va al SID per l'elaborazione finale.

Regolazione della velocità di attacco e di decadimento

I valori che rappresentano le velocità di attacco e di decadimento vengono immagazzinati nel sesto registro di ogni gruppo di voce. Il valore della velocità di attacco va nel nibble superiore e il valore della velocità di decadimento va nel nibble inferiore. Un nibble può immagazzinare valori da 0 a 15. La figura 7.10 mostra quanto tempo impiega il suono perché la sua intensità raggiunga il massimo partendo da 0, per i 16 diversi valori della velocità di attacco. Ad esempio, se il valore del nibble è 12, il tempo di attacco sarà di circa un secondo.
La figura 7.11 mostra la velocità di decadimento per le 16 diverse regolazioni, in termini del tempo impiegato dalla nota per passare dall'intensità massima all'intensità nulla. Il tempo necessario per raggiungere un dato livello di sostegno si baserà su queste velocità.
Per esempio, poniamo che il livello di sostegno sia l'80% del volume massimo e fissiamo il valore di decadimento a 6. Usando questi valori, ci vorrà il 20% di 0.199, cioè circa 0.04, secondi, per far passare il volume dal suo valore massimo al livello di sostegno.
Una volta ottenuti i valori per le velocità di attacco e di decadimento, dovete determinare il valore da inserire nel registro. Moltiplicate il valore di attacco per 16 e poi sommate il risultato al valore del decadimento. Ad esempio, supponiamo che 12 sia il valore di attacco e 6 il

Velocità di attacco

Valore del nibble	Secondi necessari per passare dal volume 0 al volume massimo	Valore del nibble	Secondi necessari per passare dal volume 0 al volume massimo
0	.002	8	.098
1	.008	9	.244
2	.016	10	.489
3	.023	11	.782
4	.037	12	.978
5	.055	13	2.933
6	.066	14	4.889
7	.078	15	7.822

FIG. 7.10. Le 16 velocità in SID selezionate dal nibble superiore del sesto registro di una voce.

Velocità di decadimento

Valore del nibble	Secondi necessari per passare dal volume 0 al volume massimo	Valore del nibble	Secondi necessari per passare dal volume 0 al volume massimo
0	.006	8	.293
1	.023	9	.733
2	.047	10	1.467
3	.070	11	2.347
4	.111	12	2.933
5	.164	13	8.800
6	.199	14	14.667
7	.235	15	23.467

FIG. 7.11. Le 16 velocità di decadimento del SID, selezionate dal nibble inferiore del sesto registro di ogni voce.

Livelli di sostegno

Valore del nibble	Percentuale del volume di picco	Valore del nibble	Percentuale del volume di picco
0	0.0	8	53.3
1	6.7	9	60.0
2	13.3	10	66.7
3	20.0	11	73.3
4	26.7	12	80.0
5	33.3	13	86.7
6	40.0	14	93.3
7	46.7	15	100.0

FIG. 7.12. I 16 livelli di sostegno dal SID, selezionati dal nibble superiore del settimo registro di ogni voce.

valore di decadimento della voce 1. 12 × 6 = 192 e sommando 6 si ottiene 198. Così voi inserite il valore 198 nel registro di attacco/decadimento in SID + 5.

Regolazione del sostegno e del rilascio

I valori che rappresentano il livello di sostegno e la velocità di rilascio sono immagazzinati nel settimo registro di ogni gruppo. Il nibble superiore contiene il valore di sostegno e quello inferiore il valore di decadimento.

Ai livelli di sostegno viene assegnata una percentuale del volume massimo. La figura 7.12 mostra le percentuali per i 16 valori possibili del nibble. Ad esempio, assegnare 9 come livello di sostegno significa che il suono scenderà del 60% rispetto al suo volume massimo. Assegnare il valore 15 significa mantenere il volume al suo massimo valore.

Le velocità di rilascio sono indicate nella figura 7.13. Questo schema è simile a quello della figura 7.11, che mostra le velocità di decadimento. I tempi che vengono elencati indicano quanto impiega un suono a scendere dal volume massimo a 0. Il tempo che effettivamente un suono impiegherà a scendere dal livello di sostegno al volume 0 è basato su questa velocità. Ad esempio, se il livello di sostegno è il 50% del volume massimo, e sceglierete il valore 10 per il rilascio, il volume impiegherà il 50% di 1.467, cioè 0.733, secondi per scendere a 0.

Una volta scelti i valori per il sostegno e il rilascio, moltiplicate il valore di sostegno per 16 e sommate al risultato il valore di rilascio. Questo

Velocità di rilascio

Valore del nibble	Secondi necessari per passare dal volume di picco al volume nullo	Valore del nibble	Secondi necessari per passare dal volume di picco al volume nullo
0	.006	8	.293
1	.023	9	.733
2	.047	10	1.467
3	.070	11	2.347
4	.111	12	2.933
5	.164	13	8.800
6	.199	14	14.667
7	.235	15	23.467

FIG. 7.13. Le 16 velocità di rilascio del SID, selezionate dal nibble inferiore del settimo registro di ogni voce.

sarà il numero da inserire nel settimo registro. Ad esempio, supponete di scegliere 3 come valore di sostegno e 11 come valore di rilascio per la voce 2. 3 × 16 = 48; sommando 11 si ottiene 59, che è il valore da inserire nelregistro SID + 13.

ATTIVARE E DISATTIVARE IL SUONO

Il quinto registro di ogni gruppo di voce è un controllore della forma d'onda. Come abbiamo già visto, il suo nibble superiore è usato per selezionare una forma d'onda. Il bit 0 di questi registri è usato per attivare o disattivare il suono, inviando un segnale di innesco (*trigger*) per il generatore di inviluppo della voce. Il bit viene definito "bit di porta".
Dare valore 1 a un bit di porta determina l'inizio di un ciclo ADSR del generatore dell'inviluppo della voce. Il volume raggiunge il massimo dal valore 0 e poi si abbassa fino al livello di sostegno e vi rimane finché il bit di porta viene regolato a 0 di nuovo. Quando questo succede, esso innesca l'azione di rilascio e il volume si abbassa a 0.

Forma d'onda	Inserire questo valore per attivare	Inserire questo valore per rilasciare
Triangolare	17	16
Dente di sega	33	32
Impulso	65	64
Rumore	129	128

FIG. 7.14. Valori da assegnare nel quinto registro di una voce per innescare o rilasciare l'inviluppo ADSR mentre si seleziona la forma d'onda.

Quando scrivete programmi sonori in Basic è una buona idea combinare la scelta della forma d'onda con l'innesco del generatore d'inviluppo. Ad esempio, assegnando a SID + 4 il valore 17, si selezionerà la forma d'onda triangolare dell'onda e si avvierà un ciclo di ADSR. Assegnando a SID + 4 il valore 16 si manterrà la forma triangolare dell'onda e si inizierà la parte di rilascio del ciclo ADSR. La figura 7.14 mostra i valori da assegnare per innescare e rilasciare un suono.

IL CONTROLLO DEL VOLUME GENERALE

Ritorniamo un attimo indietro. SID ha tre voci. Ogni voce ha il suo oscillatore e il suo generatore di forma d'onda, che producono forme d'onda a frequenze regolabili. Questi segnali vanno al modulatore d'ampiezza della voce, dove il volume viene modificato. Ogni voce usa un generatore di inviluppo per controllare il suo modulatore di ampiezza.

I segnali delle tre voci vanno poi a un controllo globale del volume. Questo meccanismo miscela le voci e regola il volume generale di uscita del SID. A volte una voce dovrà fare un giro più lungo e passare per un filtro, prima di raggiungere il controllo del volume generale, ma per ora non dovete preoccuparvene.

I bit 0-3 del registro SID + 24 regolano il volume generale, che può essere fissato a qualsiasi valore compreso fra 0 e 15. Inserendo 15 si ottiene il volume massimo, mentre inserendo 0 non si ottiene output.

LE FREQUENZE DELLE NOTE MUSICALI

La maggior parte della nostra cultura musicale è basata su scale articolare su dodici suoni: do, do#, re, re#, mi, fa, fa#, sol, sol#, la, la#, si; ovvero, per usare la notazione anglosassone, C, C#, D, D#, E, F, F#, G, G#, A, A#, B: tra una nota e la successiva vi è un intervallo fisso, che si definisce di *semitono*. Una successione di 12 semitoni costituisce un'ottava. Quando vi spostate da un'ottava ad un'altra le frequenze raddoppiano. Ciò significa che se una nota la (A) in un'ottava ha una frequenza di 440 hertz, la nota la (A) nell'ottava superiore avrà una frequenza di 880 hertz.
La frequenza di una nota è la radice dodicesima di 2 moltiplicata per la frequenza della nota precedente (cioè della nota alla distanza di un semitono discendente). In questo modo, dopo 12 note (un'ottava), la frequenza raddoppia. In un'intonazione standard, la nota la (A) nella quarta ottava ha il valore di 440 hertz. Una volta noto quel valore, si possono stabilire le altre frequenze.
L'appendice M elenca le frequenze in hertz per le 8 ottave delle note musicali, basate sull'intonazione standard. Elenca inoltre il valore di frequenza del SID per ogni nota, e spezza quel valore in un byte alto e uno basso.
Supponiamo che la voce 1 produca una nota do (C) nella quarta ottava (il do centrale). La nostra appendice ci dice che quella nota ha una frequenza di 261.6 hertz. Assegnando 16 a SID + 1 e 195 a SID, la voce 1 produrrà un suono di quella altezza.

FINALMENTE, UN PO' DI MUSICA

Ora siete pronti a mettere in pratica la vostra conoscenza di SID. La figura 7.15 mostra il listato del programma "Qualche suono". Inseritelo, salvatelo e poi fatelo girare. Esso usa la voce 1 per suonare una scala.
Consideriamo il programma. Il primo segmento pulisce lo schermo e inizializza due variabili: l'indirizzo iniziale del SID e il fattore usato per convertire le frequenze in hertz nei valori di frequenza del SID. Il segmento successivo inizializza i valori di attacco, decadimento, soste-

```
1000 REM *** QUALCHE SUONO ***
1010 :
1020 :
1030 REM ** INIZIALIZZA SCHERMO E VARIABILI
1040 :
1050 PRINT "♥";
1060 SID = 54272
1070 CNF = .0609592
1080 :
1090 :
1100 REM ** INIZIALIZZA INVILUPPO ADSR
1110 :
1120 ATK = 0                  :REM RAPIDO
1130 DKY = 0                  :REM RAPIDO
1140 AD = ATK*16 + DKY        :REM COMBINA
1150 POKE SID+5, AD           :REM REGOLA
1160 :
1170 SST = 15                 :REM VOL MAX
1180 RLS = 0                  :REM VELOCE
1190 SR = SST*16 + RLS        :REM COMBINA
1200 POKE SID+6, SR           :REM REGOLA
1210 :
1220 :
1230 REM ** INIZIALIZZA DURATA E VOLUME GENERALE
1240 :
1250 DUR = 1/4            :REM IN SECONDI
1260 VOL = 15             :REM VOL MAX
1270 POKE SID+24, VOL     :REM REGOLA
1280 :
1290 :
1300 REM ** FISSA FORMA D'ONDA E FREQUENZA
1310 :
1320 WAVFRM = 16              :REM TRIANGOLARE
1330 :
1340 READ FRQ                 :REM IN HERTZ
1350 IF FRQ = 0 THEN 1590
1360 FRQ = INT(FRQ/CNF)       :REM CONVERSIONE
1370 FHI = INT (FRQ/256)      :REM BYTE ALTO
1380 FLO = FRQ - FHI*256      :REM BYTE BASSO
1390 POKE SID, FLO            :REM REGOLA
1400 POKE SID+1, FHI
1410 :
1420 DATA 261.6, 293.7, 329.6, 349.2
1430 DATA 392.0, 440.0, 493.9, 523.3, 0
1440 :
1450 :
1460 REM ** SUONA LA NOTA, POI RITORNA
1470 :
1480 POKE SID+4, WAVFRM + 1
1490 :
1500 FOR TM = 1 TO (DUR * 700)
1510 NEXT TM
1520 :
1530 POKE SID+4, WAVFRM
1540 GOTO 1340
1550 :
1560 :
1570 REM ** PULISCE TUTTO E FINISCE
1580 :
1590 POKE SID+24, 0        :REM VOLUME SPENTO
1600 :
1610 END
```

FIG. 7.15. Listato del programma "Qualche suono".

gno e rilascio della nota. Le note raggiungeranno velocemente il volume massimo, aspetteranno che il bit di porta venga portato a zero e poi scenderanno al volume 0.
Poi viene regolato il volume generale. Scegliamo inoltre una durata per ogni nota: un quarto di secondo. Questo è il tempo che lasceremo suonare la nota prima di iniziare la fase di rilascio del ciclo ADSR.
Il segmento successivo legge le frequenze dalle istruzioni DATA e le converte in valori da assegnare nei registri di frequenza SID e SID + 1, secondo le formule che abbiamo visto nelle pagine precedenti. Il programma terminerà quando verrà letta una frequenza pari a 0.
Abbiamo fissato l'inviluppo ADSR, il volume generale e la frequenza. Adesso è ora di far suonare la nota. La linea 1480 assegna a SID + 4 un valore che seleziona la forma d'onda e innesca il generatore di inviluppo. Il volume raggiunge il valore massimo, decade al livello di sostegno e vi rimane mentre un ciclo di ritardo scandisce il tempo. La linea 1530 avvia la fase di rilascio e il volume si abbassa a 0. Poi ritorna a considerare un'altra nota.
Ora tocca a voi! Divertitevi con questo programma. Cambiate le frequenze, l'inviluppo ADSR, il volume generale, la forma d'onda, insomma tutto ciò che volete. Non ci sono formule magiche per i suoni, bisogna soltanto provare.

CHE COSA ABBIAMO IMPARATO

In questo capitolo abbiamo imparato a fare emettere suoni al Commodore 64. Vediamo che cosa abbiamo considerato:
- Suoni, vibrazioni, onde
- Frequenze, ampiezza, forma d'onda
- Le 3 voci del SID ed i meccanismi che le generano: l'oscillatore, il generatore di forma d'onda, il generatore d'inviluppo e il modulatore di ampiezza
- Lo schema generale dei 29 registri del SID
- Come regolare la frequenza di una voce, la forma d'onda, la larghezza dell'impulso e l'inviluppo ADSR
- Come attivare o disattivare una voce innescando il suo generatore d'inviluppo
- Come regolare il volume generale
- Come determinare le frequenze delle note musicali
- Come usare tutte queste informazioni in un programma che crei suoni

ESERCIZI

Test

1. Un'onda sonora può essere descritta in 3 modi: tramite la sua .... , la sua .... e la sua ....
2. Il SID ha .... voci separate.
3. I registri da SID + 7 a SID + 13 controllano la voce ....
4. Assegnando a SID il valore 16 e a SID + 1 il valore 39, attribuiamo alla voce .... una frequenza di .... hertz.
5. Assegnando 1 al bit 7 di SID + 18, selezioniamo la forma d'onda .... per la voce ....
6. Per dare alla voce 3 una larghezza d'impulso del 20%, dobbiamo assegnare a SID + 17 il valore .... e a SID + 16 il valore ....
7. Se il valore della velocità di attacco di una voce è 3, impiegherà ...... secondi per andare da 0 al volume massimo.
8. Per dare alla voce 1 un livello di sostegno che è il 40% del volume massimo e la minima velocità di rilascio possibile, si deve assegnare il valore .... in SID + ...
9. Il bit 0 del quinto registro di ogni gruppo di registri di voce è usato per innescare il generatore di .... di quella voce.
10. Il volume generale del SID è regolato dai quattro bit inferiori del registro ....
11. Se una nota do (C) della settima ottava ha una frequenza di 2093 hertz, una nota do (C) dell'ottava ottava avrà una frequenza di ... hertz.

Esercizi di programmazione

1. Cambiate il programma "Sirena minima" in modo che sembri qualcosa di spaziale.
2. Cambiate il programma "Qualche suono" in modo che corra su e giù per la scala finché non viene premuto un tasto.
3. Cambiate il programma "Qualche suono" in modo che entri anche la voce 2. Fate che la voce 2 emetta suoni diversi rispetto alla voce 1.

Risposte al test

Come sempre, tenete presente che potreste aver dato delle risposte migliori delle mie!

1. Frequenza (altezza), ampiezza (volume), forma d'onda (timbro).
2. Tre.
3. 2
4. 1; 10000
5. Rumore; 3
6. 3; 51
7. 0.023
8. 111; 6
9. Inviluppo
10. SID + 24 (54296)
11. 4186

Soluzioni possibili agli esercizi di programmazione

1. Caricate il programma "Sirena minima" poi inserite queste linee:

```
1000 REM *** RANE DA MARTE ***
1020 POKE 54278,162    :REM FISSA IL SUSTAIN
1030 POKE 54276,17     :REM NOTA ACCESA
1040 FOR N = 1 TO 30   :REM SIRENA
1050 :    POKE 54273, 1 + N*8
1051 :    POKE 54273, 1+N
1052 :    POKE 54273, 50 - N
```

2. Caricate il programma "Qualche suono" poi inserite queste linee:

```
1000 REM *** ROLLER COASTER ***
1250 DUR = 1/50          :REM IN SECONDI
1350 IF FRQ = 0 THEN RESTORE: GOTO 1340
1430 DATA 392.0, 440.0, 493.9, 523.3
1433 DATA 523.3, 493.9, 440.0, 392.0
1436 DATA 349.2, 329.6, 293.7, 261.6, 0
1513
1515 GET KP$
1517 IF KP$ <> "" THEN 1590 :REM END IT
```

3. Caricate il programma "Qualche suono" poi inserite queste linee:

```
1000 REM *** SUONI A DUE VOCI ***
1155 POKE SID+12, AD        :REM FISSA VOCE 2
1205 POKE SID+13, SR        :REM FISSA VOCE 2
1963 V2FAC = 2↑(5/12)      :REM ARMONIA?
1365 FRQ(2) = FRQ * V2FAC :REM FREQ VOCE 2
1402 FHI(2) = INT(FRQ(2)/256)  :REM V2
1404 FLO(2) = FRQ(2) - FHI(2)*256
1406 POKE SID+7, FLO(2)    :REM VOCE 2 BASSA FREQ
1408 POKE SID+8, FHI(2)    :REM VOCE 2 ALTA FREQ
1485 POKE SID+11, WAVFRM + 1
1535 POKE SID+11, WAVFRM
```

CAPITOLO OTTO

Far della musica divertente

Nel capitolo precedente abbiamo considerato il SID, l'integrato per la generazione dei suoni del Commodore 64. Ora utilizzeremo quelle conoscenze per fare musica. Insegneremo al computer a leggere le note e a immagazzinare le informazioni in una matrice d'esecuzione. Poi suoneremo le note tramite una delle voci del SID. Infine estenderemo queste tecniche all'uso di tutte e tre le voci.

LEGGERE LA MUSICA

Nel programma "Suonare" dell'ultimo capitolo, avevamo specificato le note musicali attraverso la loro frequenza. Il programma usava quel valore per calcolare le regolazioni del SID. Semplifichiamo le cose, considerando un programma che suoni le note specificate dai nomi delle lettere C, G, H ecc., (secondo l'uso anglosassone) e dal numero dell'ottava. Servirà, nel programma, una tabella di riferimento simile all'appendice M. Poi potremo ottenere che il programma legga una nota indicata da una lettera e da un'ottava, controlli il valore della frequenza nella tavola e usi quel valore per regolare i registri del SID. Ma l'appendice è abbastanza lunga. Vedremo una scorciatoia per evitare di digitarla tutta.
Nel capitolo precedente ho detto che la frequenza raddoppia, quando ci si sposta di un'ottava. Ad esempio, una nota la (A) nella quarta ottava ha una frequenza di 440 hertz, che è il doppio della frequenza (220 hertz) del la (A) nella terza ottava. Possiamo utilizzare questo fatto.

Possiamo costruire una tabella di riferimento che contiene i valori della frequenza del SID per le 12 note nell'ottava più alta, l'ottava 7. Quando il programma legge una nota, guarderà a quante ottave di distanza si trovi rispetto all'ottava più alta. Poi dividerà il valore di riferimento per 2, per ogni ottava di differenza, e arrotonderà il risultato finale al numero intero più vicino. Una volta ottenuto questo valore di frequenza, dovremo solo dividerlo per 25. La parte intera del risultato è il byte più alto del valore di frequenza, il resto è il byte più basso.
Ad esempio, supponiamo che il programma legga una nota che è un fa# (F#) della seconda ottava, ed è quindi a otto ottave di distanza rispetto all'ottava più alta. Il valore di frequenza del SID per un fa# (F#) della settima ottava è 48557. Dividendo quel valore per 2 si ottiene 24278.5. Dopo altre 4 divisioni, si arriva al valore 1517.4062, che viene arrotondato a 1517. Dividendo questo valore per 256, si ottiene 5 per il byte più alto e 237 per quello più basso. Controllando nell'appendice M, noterete che questo metodo ci ha dato il valore corretto. La figura 8.1 mostra i nomi delle lettere delle 12 note nella settima ottava, con le loro frequenze espresse in hertz e il corrispondente valore di frequenze del SID.
Per creare musica occorre ora specificare il nome di una nota e il numero dell'ottava per ogni nota. Si può farlo con le stringhe. Ad esempio, si può rappresentare un sol# (G#) della quinta ottava come G#-5.

Nota	Frequenza in hertz	Regolazione della frequenza del SID
Do	2093.0	34334
Do#	2217.5	36377
Re	2349.3	38539
Re#	2489.0	40831
Mi	2637.0	43258
Fa	2793.8	45831
Fa#	2960.0	48557
Sol	3136.0	51444
Sol#	3322.4	54502
La	3520.0	57743
La#	3729.3	61177
Si	3951.1	64815

FIG. 8.1. Le 12 note della settima ottava, da usare come ottava di riferimento.

Un programma può usare le funzioni di stringa per estrarre il nome e l'ottava della nota dai dati immagazzinati in questo modo.

Durata delle note

Nel programma "Fare un po' di musica", ogni nota durava un certo tempo. Era abbastanza noioso. Possiamo però inserire nelle istruzioni DATA del programma un numero che indichi la durata per ogni nota. Prendiamo spunto dalla notazione musicale e creiamo una "durata standard", che chiameremo movimento. Poi la durata di ogni nota sarà espressa con un numero di movimenti. Ad esempio, potete rappresentare una nota fa (F) nella terza ottava, che dura 4 movimenti, come una stringa e un intero:

```
F-3,4
```

Come può un programma far durare una nota per 2 movimenti e un'altra nota per 3? Ci sono vari modi per farlo. Uno dei più flessibili è usare quelle che io chiamo matrici di esecuzione.

MATRICI DI ESECUZIONE

Una matrice di esecuzione contiene un valore di SID per ogni movimento di un brano musicale. Un programma può avere diverse matrici di esecuzione. Una matrice potrebbe contenere i byte bassi per il valore di frequenza della voce 1, e un'altra potrebbe contenere i byte alti. Una terza matrice potrebbe contenere i valori per il registro di attacco/decadimento della voce 1.
Quando il programma deve eseguire le note, entrerà semplicemente in un ciclo di movimenti. Ogni volta che entra nel ciclo, le regolazioni del SID per quel movimento saranno estratti dalle matrici di esecuzione e inseriti al posto giusto con le istruzioni POKE.
Ci sarà un piccolo ritardo, pari alla durata di un movimento, e poi il programma riprenderà il ciclo per il movimento successivo.
Una nota della durata di un movimento avrà un'entrata in ogni matrice di esecuzione; una nota con una durata maggiore avrà tante entrate quanti sono i movimenti. Ad esempio, supponiamo che una delle nostre matrici di esecuzione immagazzini valori per il byte alto del valore della frequenza della voce 1. Se la prima nota di un brano è un re (D) della quarta ottava che dura tre movimenti e la seconda nota è un fa# (F#)

```
1000 REM *** LEGGI LA MUSICA ***
1010 :
1020 :
1030 REM ** INIZIALIZZA SCHERMO E VARIABILI
1040 :
1050 PRINT "J";         :REM PULISCE SCHERMO
1060 PRINT "QQQQQQQQQQQQSTO LEGGENDO";
1070 :
1080 SID = 54272        :REM CHIP SONORO
1090 :
1100 :
1110 REM ** PREDISPONE LE MATRICI DI RIFERIMENTO
1120 :
1130 DIM SBN(11), NM$(11) :REM BASATO SULLE
1140 FOR N = 0 TO 11       :REM NOTE DELLA
1150 :   READ SBN(N)       :REM OTTAVA
1160 :   READ NM$(N)       :REM PIU' ALTA
1170 NEXT N
1180 :
1190 DATA 34334, C,  36377, C#
1200 DATA 38539, D,  40831, D#
1210 DATA 43258, E,  45831, F
1220 DATA 48557, F#, 51444, G
1230 DATA 54502, G#, 57743, A
1240 DATA 61177, A#, 64815, B
1250 :
1260 :
1270 REM ** LEGGE LA MUSICA E LA MEMORIZZA IN MATRICI
1280 :
1290 DIM LFP(200), HFP(200)
1300 :
1310 EVENT = 1
1320 :
1330 READ NC$
1340 PRINT ".";
1350 IF NC$ = "XXX" THEN 1670
1360 :
1370 GOSUB 2050 :REM CONVERSIONE A POKE
1380 :
1390 READ DUR
1400 FOR N = 1 TO DUR
1410 :   LFP(EVENT) = LFP
1420 :   HFP(EVENT) = HFP
1430 :   EVENT = EVENT + 1
1440 NEXT N
1450 :
1460 GOTO 1330
1470 :
1480 :
1490 REM ** LA MUSICA: NOTA-OTTAVA, DURATA
1500 :
1510 DATA B-4, 4, D-5, 4, C-5, 8
1520 DATA RES, 1, B-4, 4, D-5, 4
1530 DATA A-4, 8, RES, 1, B-4, 4
1540 DATA D-5, 4, C-5, 4, B-4, 2
1550 DATA C-5, 2, D-5, 4, A-4, 4
1560 DATA G-4, 8, RES, 2, B-4, 4
1570 DATA G-4, 4, C-5, 8, RES, 1
1580 DATA B-4, 4, G-4, 4, A-4, 8
1590 DATA RES, 1, B-4, 4, D-5, 4
1600 DATA C-5, 8, RES, 1, B-4, 2
1610 DATA C-5, 2, D-5, 4, A-4, 4
1620 DATA G-4,10, XXX
1630 :
```

```
1640 :
1650 REM ** FISSA ADSR, VOLUME, FORMA D'ONDA
1660 :
1670 ATK = 0            :REM ATTACCO RAPIDO
1680 DKY = 0            :REM DECADIMENTO RAPIDO
1690 AD = ATK*16 + DKY
1700 POKE SID+5, AD
1710 :
1720 SST = 15           :REM SOSTEGNO FORTE
1730 RLS = 0            :REM RILASCIO RAPIDO
1740 SR = SST*16 + RLS
1750 POKE SID+6, SR
1760 :
1770 VLM = 15           :REM VOLUME MASSIMO
1780 POKE SID+24, VLM
1790 :
1800 WVFRM = 16         :REM ONDA TRIANGOLARE
1810 :
1820 :
1830 REM ** SUONA LA MUSICA, POI FINISCI
1840 :
1850 PRINT "J";
1860 BEATLNGTH = 10
1870 :
1880 FOR N = 1 TO (EVENT - 1)
1890 :    POKE SID+1, HFP(N)
1900 :    POKE SID, LFP(N)
1910 :
1920 :    POKE SID+4, WVFRM + 1   :REM ACCESO
1930 :    FOR TM = 1 TO BEATLNGTH
1940 :    NEXT TM
1950 NEXT N
1960 :
1970 POKE SID+4, 0   :REM FORMA D'ONDA SPENTA
1980 POKE SID+24,0   :REM VOLUME SPENTO
1990 END
2000 :
2010 :
2020 :
2030 REM ** CONVERSIONE DI STRINGHE NOTA-OTTAVA
2031 REM    IN CODICI DI POKE BASSO E ALTO
2040 :
2050 IF NC$ = "RES" THEN HFP = 0 :  LFP = 0 : RETURN
2060 :
2070 NT$ = LEFT$(NC$, LEN(NC$) - 2)
2080 FOR REF = 0 TO 11
2090 :    IF NT$ = NM$(REF) THEN NT = REF : REF = 11
2100 NEXT REF
2110 :
2120 OCT = VAL(RIGHT$(NC$,1))
2130 :
2140 FST = 2 ↑ (7 - OCT)
2150 FST = SBN(NT) / FST
2160 HFP = INT (FST/256)
2170 LFP = INT (FST - 256*HFP)
2180 :
2190 RETURN
```

FIG. 8.2. Listato del programma "Leggi la musica".

della quinta ottava che dura due movimenti, la matrice inizierà con questi 5 valori:

```
HF(1) = 18
HF(2) = 18
HF(3) = 18
HF(4) = 47
HF(5) = 47
```

Utilizzando le matrici di esecuzione si hanno molti vantaggi. Dato che i valori del SID vengono calcolati prima che ogni nota venga suonata, le note possono susseguirsi fluidamente, senza ritardi dovuti alla lunghezza dei calcoli. E dato che l'unità di tempo fondamentale è il movimento, è facile avere voci diverse che suonano note di diversa lunghezza, come vedremo più avanti in questo capitolo.
È ora di passare dalla teoria alla pratica: vediamo come vengono usati in un programma la lettura di una nota e le matrici di esecuzione.

UN PROGRAMMA CHE LEGGE MUSICA E LA SUONA

La figura 8.2 mostra il listato del programma "Leggi la musica", che usa le idee appena discusse. Leggetelo, inseritelo nel computer, salvatelo e fatelo girare. Se volete risentirlo senza aspettare che la musica venga riletta nelle matrici di esecuzione, inserite questo comando:

```
GOTO 1670
```

La melodia suonata da questo programma è un vecchio motivo inglese chiamato "Shepherd's hey".
Consideriamo il programma nei suoi dettagli. Le linee 1050-1080 puliscono lo schermo, stampano il simbolo di inizio scrittura (prompt) e fissando l'indirizzo iniziale del SID. La matrice SBN contiene i 12 valori di frequenza del SID per la settima ottava e la matrice NM$ contiene i 12 nomi che corrispondono alle note.
Il segmento successivo legge la nota e riempie le matrici di esecuzione. In questo caso, avete una matrice di esecuzione che conterrà i byte alti. La linea 1330 legge una stringa nota/ottava e poi la linea 1340 manda un segnale di retroazione allo schermo.
La linea 1350 cerca la stringa che segnala la fine dei dati nota/ottava. Se la trova, la lettura delle note termina e il programma prosegue a fissare l'inviluppo ADSR. La linea 1370 salta a una subroutine che prende la

stringa nota/ottava e calcola gli appropriati byte alto e basso per un valore di frequenza del SID. Vediamo ora come funziona questa subroutine.

Come si decodifica la stringa nota/ottava

La linea 2050 prima di tutto cerca il valore speciale RES della stringa. RES indica una pausa musicale: una nota di silenzio. Assegnare il valore 0 ai registri di frequenza del SID è un modo per creare silenzio.
La linea 2070 prende il nome della nota dalla stringa. Poi le linee 2080-2100 cercano se il nome della nota corrisponde a qualcuno dei nomi contenuti nella matrice NM$ di riferimento. Quando c'è una corrispondenza, il programma immagazzina il numero della nota nella variabile NT. Questo numero verrà usato per estrarre dalla matrice SBN l'appropriata frequenza di riferimento del SID.
La linea 2120 estrae i numeri dell'ottava dalla stringa. La linea 2140 usa questo numero per calcolare il valore per cui va diviso il valore della frequenza di riferimento. La linea 2150 effettuerà la divisione. Infine le istruzioni 2160-2170 calcolano i byte alto e basso che daranno questo valore. La conversione a questo punto è completa e la subroutine ritorna alla linea 1380.

Come si riempiono le matrici di esecuzione

Ora occorre procedere alla somma delle matrici di esecuzione. Ricordatevi che dovete inserire un'informazione per ogni movimento. La linea 1390 legge la durata della nota espressa come numero di movimenti. Le linee 1400-1440 usano questo valore per controllare un ciclo che "impacchetta" le due matrici di esecuzione. Il corpo del ciclo verrà eseguito una volta per ogni movimento della nota. Ogni volta che si entra nel ciclo, i byte alto e basso del valore della frequenza della nota si inseriscono nelle matrici e poi il numero del movimento aumenta di 1.
Pensate che questo discorso sia confuso? Affrontiamolo da un altro punto di vista. In realtà quello che stiamo facendo è creare una copia delle regolazioni di una nota.
Vengono effettuate tante copie quanti sono i movimenti della nota. Quando bisogna eseguire il pezzo, il programma prenderà un movimento alla volta, i valori del SID.

La musica

Le linee 1510-1620 memorizzano la musica. La stringa XXX segnala la fine dell'informazione. Se volete cambiate canzone dovete soltanto cambiare queste istruzioni. Potete scegliere canzoni da spartiti o crearne. Se scegliete canzoni da spartiti, dovete sapere leggere la musica.

ADSR e forma d'onda

Le linee 1670-1750 inseriscono i valori di attacco, decadimento, sostegno e rilascio per la voce 1. Le linee 1770-1780 regolano il volume generale e la linea 1800 seleziona la forma d'onda che verrà usata. Ho scritto queste istruzioni in modo che risulti facile effettuare cambiamenti.
Infine, tutto è pronto.
Il sipario si alza e il direttore d'orchestra si prepara a iniziare il concerto (linee 1850-1860). Il ciclo nelle linee 1880-1950 suona la musica; un movimento alla volta. Ogni volta che si entra nel ciclo, viene immagazzinato il valore della frequenza di quel movimento. Poi la linea 1920 innesca il modulatore d'ampiezza che inizia il ciclo ADSR.
Per semplicità a questo punto ho adottato un trucco. Il ciclo di esecuzione non innesca mai la fase di rilascio dell'inviluppo.
Le note si legano insieme. Provate a far girare il programma inserendo questa istruzione:

```
1945 :    POKE SID+4, WAVFRM :REM RILASCIO
```

Notate come vengono staccate le note più lunghe di un movimento se si innesca la fase di rilascio alla fine di ogni movimento. Esiste un modo per evitare sia che le note si leghino, sia che si stacchino? Sì, vedrete questa tecnica più avanti. Infine, le linee 1970-1980 disattivano i controlli della forma d'onda e del volume generale.
Ora tocca a voi. Fate in modo che questo programma suoni un motivo diverso o fatelo suonare a diversa velocità. Guardate che cosa succede quando due o più note della stessa altezza si susseguono. Se non sapete leggere la musica, trovate un amico che la sappia leggere e inventate delle note che insieme costituiscano una piccola composizione oppure inserite le istruzioni elencate in figura 8.3.

```
1510 DATA G-4, 4, E-4, 4, G-4, 4
1520 DATA C-5, 4, D-5, 2, 3-5, 2
1530 DATA D-5, 4, B-4, 2, C-5, 2
1540 DATA B-4, 4, RES, 1, G-4, 4
1550 DATA E-4, 4, G-4, 4, C-5, 4
1560 DATA D-5, 2, E-5, 2, D-5, 4
1570 DATA B-4, 4, C-5, 4, XXX
```

FIG. 8.3. Variazioni da apportare al programma "Leggi la musica" per eseguire un motivo diverso.

TRE VOCI

Si possono apportare altri due perfezionamenti ai programmi strutturati come "Leggi la musica". Primo, potete mettere in azione altre due voci di SID. Secondo, potete trovare un metodo per rendere ogni nota più distinta, senza legarla o staccarla.
Questi due miglioramenti sono facilmente ottenibili con le matrici di esecuzione. Guardiamo il primo perfezionamento: in "Leggi la musica" avete immagazzinato l'informazione sulla frequenza della voce 1 per ogni movimento.
Aggiungerete una informazione simile per la frequenza per le altre due voci. Immagazzinerete l'informazione in matrici bidimensionali.
Essi avranno la seguente forma:

NOMEMATRICE (voce, battuta)

Ecco alcuni esempi che utilizzano i nomi delle matrici di "Leggi la musica":

LFP (1,20) contiene il byte basso del valore della frequenza della voce 1 per il ventesimo movimento
HFP (3,80) contiene il byte alto del valore della frequenza della voce 3 per l'ottantesimo movimento
HFP (2,1) contiene il byte alto del valore della frequenza della voce 2 per il primo movimento

Per quanto riguarda il secondo perfezionamento, si vuole rendere ogni nota più distinta. Nel programma "Leggi la musica", il ciclo d'esecuzione si limitava a innescare un ciclo ADSR e non affrontava mai la fase di rilascio, ma aggiungendo uno stadio di rilascio a ogni movimento si stacca troppo.
Una cosa che si può fare è innescare una fase di rilascio all'ultimo

movimento di una nota. Cioè, se una nota dura 4 movimenti, ognuno dei primi tre innescherà un ciclo ADSR e l'ultimo inizierà la fase di rilascio. Non è una soluzione perfetta, ma abbastanza soddisfacente. Creerete una nuova matrice di esecuzione per il controllo della forma d'onda, le cui entrate (per ogni voce e per ogni movimento) saranno valori da assegnare in ogni registro di controllo della forma d'onda della voce.
Ecco un esempio: supponete che la voce 1 inizi suonando una nota che dura tre movimenti e supponete di selezionare la forma d'onda triangolare per la voce 1. Assegnate il nome WVC alla matrice di controllo dell'onda. Poi WVC(1,1) conterrà il valore 17. WVC(1,2) conterrà il valore 17. WVC(1,3) conterrà il valore 16. I valori dei primi due movimenti della nota inizieranno un ciclo ADSR. Il valore dell'ultimo movimento della nota inizierà la fase di rilascio del ciclo.

UN ESEMPIO A TRE VOCI

La figura 8.4 mostra il listato del programma "Canzone a tre voci". Inseritelo, salvatelo e fatelo girare. Cercate di confrontare questo programma con "Leggi la musica" rappresentato in figura 8.2: sono molto simili. Nella nostra discussione, mi concentrerò sulle loro differenze.

```
1000 REM *** CANZONE A TRE VOCI ***
1010 :
1020 :
1030 REM ** INIZIALIZZA SCHERMO E VARIABILI
1040 :
1050 PRINT "[illegible]";        :REM PULISCE LO SCHERMO
1060 PRINT "[illegible]STO LEGGENDO";
1070 :
1080 SID = 54272       :REM CHIP SONORO
1090 WV = 16  :REM STESSA FORMA D'ONDA PER TUTTE E TRE
1100 :
1110 :
1120 REM ** PREDISPONE MATRICI DI RIFERIMENTO
1130 :
1140 DIM SBN(11), NM$(11) :REM BASATO SU
1150 FOR N = 0 TO 11       :REM NOTE DELLA
1160 :    READ SBN(N)       :REM OTTAVA
1170 :    READ NM$(N)       :REM PIU' ALTA
1180 NEXT N
1190 :
1200 DATA 34334, C,  36377, C#
1210 DATA 38539, D,  40831, D#
1220 DATA 43258, E,  45831, F
1230 DATA 48557, F#, 51444, G
1240 DATA 54502, G#, 57743, A
1250 DATA 61177, A#, 64815, B
```

```
1260 :
1270 :
1280 REM ** LEGGE LA MUSICA
1281 REM    E LA MEMORIZZA NELLE MATRICI
1290 :
1300 DIM LFP(3,200), HFP(3,200), WVC(3,200)
1310 :
1320 VOICE = VOICE + 1
1330 IF VOICE = 4 THEN 1890
1340 EVENT = 1
1350 :
1360 READ NC$
1370 PRINT ".";
1380 IF NC$ = "XXX" THEN 1320
1390 :
1400 GOSUB 2440 :REM CONVERSIONE IN POKE
1410 :
1420 READ DUR
1430 FOR N = 1 TO DUR-1
1440 :    LFP(VOICE,EVENT) = LFP
1450 :    HFP(VOICE,EVENT) = HFP
1460 :    WVC(VOICE,EVENT) = WV + 1
1470 :    EVENT = EVENT + 1
1480 NEXT N
1490 IF DUR = 1 THEN 1360
1500 :
1510 LFP(VOICE,EVENT) = LFP
1520 HFP(VOICE,EVENT) = HFP
1530 WVC(VOICE,EVENT) = WV
1540 EVENT = EVENT + 1
1550 :
1560 GOTO 1360
1570 :
1580 :
1590 REM ** MUSICA: NOTA-OTTAVA, DURATA
1600 :
1610 DATA RES,    4, A-5,    4, B-5,    4
1620 DATA A-5,    4, RES,    4, A-5,    4
1630 DATA B-5,    4, A-5,    4, RES,    4
1640 DATA C-6,    4, E-6,    4, C-6,    4
1650 DATA A-5,    4, A-5,    4, B-5,    4
1660 DATA A-5,    4, XXX
1670 :
1680 DATA RES,    4, E-5,    4, E-5,    4
1690 DATA E-5,    4, RES,    4, E-5,    4
1700 DATA E-5,    4, E-5,    4, RES,    4
1710 DATA G-5,    4, G-5,    4, G-5,    4
1720 DATA E-5,    4, E-5,    4, E-5,    4
1730 DATA E-5,    4, XXX
1740 :
1750 DATA C-5,    2, D-5,    2, C-5,    2
1760 DATA A-4,    2, A-4,    2, G-4,    2
1770 DATA A-4,    4, C-5,    2, D-5,    2
1780 DATA C-5,    2, A-4,    2, A-4,    2
1790 DATA G-4,    2, A-4,    4, C-5,    2
1800 DATA D-5,    2, E-5,    2, E-5,    2
1810 DATA E-5,    2, G-5,    2, E-5,    2
1820 DATA D-5,    2, C-5,    2, A-4,    2
1830 DATA C-5,    2, A-4,    2, A-4,    2
1840 DATA G-4,    2, A-4,    4, XXX
1850 :
1860 :
1870 REM ** FISSA L'ADSR PER LE 3 VOCI
1880 :
1890 ATK = 2 : DKY = 3 :REM REGOLAZIONI
1900 AD = ATK*16 + DKY  :REM VALORE DA INSERIRE
```

```
1910 POKE SID+5, AD     :REM A-D VOCE 1
1920 :
1930 ATK =  2 : DKY = 3 :REM REGOLAZIONI
1940 AD = ATK*16 + DKY  :REM VALORE DA INSERIRE
1950 POKE SID+12, AD    :REM A-D VOCE 2
1960 :
1970 ATK =  2: DKY = 0  :REM REGOLAZIONI
1980 AD = ATK*16 + DKY  :REM VALORE DA INSERIRE
1990 POKE SID+19, AD    :REM A-D VOCE 3
2000 :
2010 SST = 6   : RLS = 6 :REM REGOLAZIONI
2020 SR = SST*16 + RLS  :REM VALORE DA INSERIRE
2030 POKE SID+6, SR     :REM S-R VOCE 1
2040 :
2050 SST = 12 : RLS = 6 :REM REGOLAZIONI
2060 SR = SST*16 + RLS  :REM VALORE DA INSERIRE
2070 POKE SID+13, SR    :REM S-R VOCE 2
2080 :
2090 SST = 15 : RLS = 7 :REM REGOLAZIONI
2100 SR = SST*16 + RLS  :REM VALORE DA INSERIRE
2110 POKE SID+20, SR    :REM S-R VOCE 3
2120 :
2130 :
2140 REM ** SUONA, POI FINISCI
2150 :
2160 PRINT "□";
2170 BEATLNGTH = 10
2180 VLM = 15           :REM VOLUME MASSIMO
2190 POKE SID+24, VLM
2200 :
2210 FOR N = 1 TO (EVENT - 1)
2220 :    POKE SID+1, HFP(1,N)
2230 :    POKE SID, LFP(1,N)
2240 :    POKE SID+8, HFP(2,N)
2250 :    POKE SID+7, LFP(2,N)
2260 :    POKE SID+15, HFP(3,N)
2270 :    POKE SID+14, LFP(3,N)
2280 :
2290 :    POKE SID+4,  WVC(1,N):REM V-1
2300 :    POKE SID+11, WVC(2,N):REM V-2
2310 :    POKE SID+18, WVC(3,N):REM V-3
2320 :
2330 :    FOR TM = 1 TO BEATLNGTH
2340 :    NEXT TM
2350 NEXT N
2360 :
2370 POKE SID+24,0  :REM VOLUME A ZERO
2380 END
2390 :
2400 :
2410 :
2420 REM ** TRASFORMA LE STRINGHE NOTA-OTTAVA
2421 REM-  IN CODICI ALTO E BASSO DA INSERIRE
2430 :
2440 IF NC$ = "RES" THEN HFP = 0 : LFP = 0 : RETURN
2450 :
2460 NT$ = LEFT$(NC$, LEN(NC$) - 2)
2470 FOR REF = 0 TO 11
2480 :    IF NT$ = NM$(REF) THEN NT = REF : REF = 11
2490 NEXT REF
2500 :
2510 OCT = VAL(RIGHT$(NC$,1))
2520 :
2530 FST = 2 ↑ (7 - OCT)
2540 FST = SBN(NT) / FST
```

```
2550 HFP = INT (FST/256)
2560 LFP = INT (FST - 256*HFP)
2570 :
2580 RETURN
```

FIG. 8.4. Listato del programma "Canzone a tre voci".

Riempire le matrici di esecuzione

Il primo cambiamento si nota nella riga 1090. Il programma inizializza una variabile di forma d'onda, essa verrà usata per riempire la matrice di esecuzione del controllo della forma d'onda. Per altro, i primi 2 moduli sono uguali: si pulisce lo schermo, lo si inizializza e si riempiono le matrici di riferimento.
Le linee 1300-1560 leggono le note e riempiono le matrici. Prima di tutto, la linea 1300 determina la dimensione delle 3 matrici di esecuzione. Due conterranno i valori della frequenza e la terza conterrà i valori di controllo della forma d'onda.
Questa parte del programma legge le note e "impacchetta" una voce alla volta nelle matrici. La pseudonota XXX segnala la fine di una delle note della voce. Il numero della voce poi si incrementa di 1. Quando raggiunge il valore 4, tutte e tre le voci sono state considerate e il programma va a regolare i valori dell'ADSR. Quando la linea 1360 legge una nota valida, il programma salta alla stessa subroutine di calcolo della frequenza usata dal programma "Leggi la musica". Questa subroutine restituisce i valori dei byte alto e basso del valore della frequenza.
Se una nota ha la durata di un movimento, entrerà nel ciclo di "impacchettamento", nelle linee 1430-1480, una sola volta. Le linee 1440-1450 inseriscono i byte alto e basso della frequenza. Poi la linea 1460 assegna alle matrici di controllo della forma d'onda il valore che innescherà l'inviluppo ADSR. La linea 1490 rimanda il programma a leggere un'altra nota.
Una nota che dura più di un movimento viene trattata diversamente. Entrerà nel ciclo nelle linee 1430-1480 una volta in meno rispetto alla sua durata in movimenti. Così, su tutti i movimenti fino all'ultimo la matrice di controllo della forma d'onda riceverà un valore che innescherà un inviluppo ADSR. Le linee 1510-1540 manipolano le matrici per l'ultimo movimento. La frequenza non viene manipolata diversamente, comunque la matrice di controllo della forma d'onda ora ottiene un valore che inizierà la fase di rilascio dell'inviluppo ADSR.

```
1000 REM *** COVENTRY CAROL ***
1610 DATA A-4,   4, A-4,   4, G#-4,  4
1620 DATA A-4,   8, C-5,   4, B-4,   8
1630 DATA A-4,   4, G#-4, 12, RES,   1
1640 DATA A-4,   4, B-4,   4, C-5,   4
1650 DATA D-5,   8, B-4,   4, A-4,  20
1660 DATA RES,   1, E-5,   4, D-5,   8
1670 DATA C-5,   4, B-4,   8, C-5,   4
1680 DATA B-4,   8, A-4,   4, G#-4, 12
1690 DATA RES,   1, A-4,   4, G#-4,  4
1700 DATA A-4,   4, D-5,   4, B-4,   8
1710 DATA C#-5, 12, XXX
1715 :
1720 DATA C-4,   8, D-4,   4, E-4,   8
1725 DATA E-4,   4, F-4,   8, F-4,   4
1730 DATA E-4,  12, RES,   1, E-4,   8
1735 DATA A-4,   4, A-4,   4, F-4,   4
1740 DATA G-4,   4, E-4,   8, D-4,   4
1745 DATA C-4,   8, RES,   1, G-4,   4
1750 DATA A-4,   8, E-4,   4, G-4,   8
1755 DATA A-4,   4, G-4,   8, E-4,   4
1760 DATA E-4,   8, D-4,   4, RES,   1
1765 DATA E-4,   4, F-4,   4, E-4,   4
1770 DATA F-4,   4, G-4,   8, E-4,  12
1775 DATA XXX
1780 :
1785 DATA A-2,   8, B-2,   4, C-3,   8
1790 DATA A-2,   4, D-3,   8, D-3,   4
1795 DATA E-3,   8, D-3,   4, RES,   1
1800 DATA C-4,   4, B-3,   4, A-3,   4
1805 DATA B-2,   8, E-3,   4, C-3,   8
1810 DATA B-2,   4, A-2,   8, RES,   1
1815 DATA C-3,   4, F-2,   8, A-2,   4
1820 DATA E-3,   8, E-3,   4, E-3,   8
1825 DATA C-3,   4, B-2,  12, RES,   1
1830 DATA C-3,   4, D-3,   4, C-3,   4
1835 DATA B-2,   8, E-3,   4, A-2,  12
1840 DATA XXX
1890 ATK =  2 : DKY = 3 :REM REGOLAZIONI
1930 ATK =  2 : DKY = 3 :REM REGOLAZIONI
1970 ATK =  2 : DKY = 0 :REM REGOLAZIONI
2010 SST =  4 : RLS = 6 :REM REGOLAZIONI
2050 SST =  9 : RLS = 6 :REM REGOLAZIONI
2090 SST = 15 : RLS = 7 :REM REGOLAZIONI
```

FIG. 8.5. Cambiamenti del programma "Canzone a tre voci" che insegnano al programma a suonare la canzone "Coventry Carol".

Regolazione degli inviluppi ADSR

Dopo la lettura delle note e dopo il riempimento delle matrici di esecuzione, è ora di regolare gli inviluppi ADSR per ogni voce. Le routine usate nelle linee 1890-2110 usano la stessa tecnica mostrata nel programma "Leggi la musica". Ecco un consiglio: le note basse hanno bisogno di livelli di sostegno maggiori per essere percepite bene come le note più acute. Questo per la conformazione delle nostre orecchie. Nel

programma la voce 1 suona le note più acute, la voce 3 le più basse e la voce 2 quelle centrali. Quindi ho attribuito alla voce 3 il livello di sostegno più alto, alla voce 1 il più basso ed alla voce 2 un livello intermedio.

Suonare

Dopo qualche ritocco conclusivo, il programma può eseguire la musica. Le linee 2160-2190 puliscono lo schermo, fissano la lunghezza di un movimento e regolano il volume generale. Poi entra in gioco il ciclo di esecuzione, che si ripeterà tante volte quanti sono i movimenti. Le linee 2220-2270 regolano i registri di frequenza per tutte e tre le voci. Poi le linee 2290-2310 estraggono i valori della nuova matrice di controllo della forma d'onda e li assegnano nel registro di controllo di ciascuna voce. Le voci operano indipendentemente; su ogni movimento dato, due voci potrebbero iniziare un'inviluppo ADSR e l'altra potrebbe iniziare la fase di rilascio.
La tecnica di avviare la fase di rilascio di una voce al suo ultimo movimento funziona bene se c'è un periodo di rilascio abbastanza lungo. Cambiate i valori di rilascio nelle linee 2010, 2050 e 2090 con valori più bassi e poi fate girare il programma. Avvertite la differenza?

Variazioni

I dati nel programma "Canzone a tre voci" sono basati sulla melodia popolare inglese "Are you Going to the Fair". La figura 8.5 rivolge il nostro saluto alla musica inglese pre-Beatles. Caricate il programma "Canzone a tre voci" e inserite le linee elencate in figura 8.5. Ora il vostro Commodore 64 suonerà la canzone "Coventry Carol".
Potete utilizzare il programma "Canzone a tre voci" per fare alcuni esperimenti. Per esempio, guardate se potete ottenere che le tre voci suonino come strumenti completamente diversi. E ricordatevi: SID può imitare gli strumenti reali, ma eccelle veramente nei suoni mai sentiti né dai legni, né dagli ottoni, né dagli archi...

CHE COSA ABBIAMO IMPARATO

Abbiamo esaminato due modi per ottenere musica dal Commodore 64. Ecco che cosa abbiamo imparato:

– Come fissare un'ottava di riferimento per facilitare la traduzione dei nomi delle note e dei numeri delle ottave nei valori di frequenza del SID
– Come usare le matrici di esecuzione per immagazzinare le informazioni sulla regolazione di frequenza del SID per ogni movimento di un brano musicale
– Come usare le matrici per implementare la musica a tre voci
– Come attivare e disattivare le voci con una matrice di esecuzione di controllo della forma dell'onda

Nel prossimo capitolo, faremo in modo che SID generi alcuni effetti musicali stimolanti.

ESERCIZI

Test

1. Se la nota la (A) della terza ottava ha una frequenza di 220 hertz, qual è la frequenza della nota la (A) della prima ottava?
2. Usando la notazione a stringa, CH-6 rappresenta una nota do diesis della .... ottava.
3. Una matrice di esecuzione può contenere valori del SID per ogni .... di una canzone.
4. Il programma "Leggere musica" immagazzina valori della .... per ogni movimento nelle matrici di esecuzione LEP(200) e HFP(200).
5. Un modo per manipolare più di una voce alla volta è quello di usare matrici di esecuzione .... dimensionali.
6. Potete evitare il legato e lo staccato delle note iniziando la fase di .... sull'ultimo movimento delle note.
7. Guardate il programma "Canzone a tre voci". Qual è il numero minimo di movimenti che una nota può avere, per poter ancora innescare la fase di rilascio?

Esercizi di programmazione

1. Cambiate il programma "Leggi la musica" in modo che ripeta, se si desidera, la musica, senza dovere costruire nuovamente le matrici di esecuzione.
2. Cambiate il programma "Canzone a tre voci" in modo che dia la possibilità all'utente di regolare il tempo musicale (la velocità).

3. Cambiate il programma "Canzone a tre voci" in modo che dia la possibilità all'utente di regolare l'altezza complessiva, in ottave.

Risposte al test

1. 55 hertz
2. sesta
3. movimento
4. frequenza
5. bi
6. rilascio
7. 2

Soluzioni possibili agli esercizi di programmazione

1. Caricate il programma "Leggi la musica" poi inserite queste linee:

```
1000 REM *** JUKE BOX ***
1030 REM ** SUONA LA MUSICA
1990 :
1991 :
1992 REM ** SUONA DI NUOVO?
1993 :
1994 PRINT "[illegible]PREMI UN ";
1995 PRINT "TASTO QUALUNQUE ENTRO 5"
1996 PRINT "[illegible]SECONDI SE ";
1997 PRINT "VUOI CHE RIPETA"
1998 :
1999 TI$ = "000000"  :REM REINIZIALIZZA IL TEMPO
2000 :
2001 GET KY$          :REM LEGGE LA TASTIERA
2002 IF KY$ <> "" THEN 1770
2003 IF VAL(TI$) < 5 THEN 2001
2004 :
2005 PRINT "[illegible]";
2006 END
2007 :
```

2. Caricate il programma "Canzone a tre voci" poi inserite queste linee:

```
1000 REM *** TEMPO REGOLABILE ***
1001 :
1002 :
1003 REM ** PRENDI IL TEMPO
1004 :
1005 PRINT "[illegible]PREMI UN TASTO";
1006 PRINT "[illegible]PER DARE IL[illegible]TEMPO : "
1007 PRINT "[illegible](1-PIU' LENTO ";
1008 PRINT "9-PIU' VELOCE)[illegible]";
1009 :
```

```
1010 GET KY$
1011 IF KY$ = "" THEN 1010
1012 IF ASC(KY$) < 49  OR  ASC(KY$) > 57 THEN 1010
1013 :
1014 PRINT "■"; KY$; "■"
1015 TEMPO = VAL(KY$)
1016 :
1017 FOR N = 1 TO 500
1018 NEXT N
2170 BEATLNGTH = (10 - TEMPO) ↑ 1.7
```

3. Caricate il programma "Canzone a tre voci" poi inserite queste linee:

```
1000 REM *** TRASPORTAOTTAVA ***
1001 :
1002 :
1003 REM ** CHIEDE DI QUANTE OTTAVE TRASPORTARE
1004 :
1005 PRINT "■■■■■■■■■■■■■■■DI QUANTE ";
1006 PRINT "OTTAVE VUOI "
1007 PRINT "■■■■■■■■■TRASPORTARE ";
1008 PRINT "(0 - 3) ? ";
1009 GET ADJ$
1010 IF ADJ$ = "" THEN 1009
1011 :
1012 IF ASC(ADJ$) < 48 OR ASC(ADJ$) > 51 THEN 1009
1013 :
1014 PRINT "■"; ADJ$; "■" :REM STAMPA
1015 ADJ = VAL (ADJ$)
1016 IF ADJ = 0 THEN 1027 :REM NON C'E' SECONDO?
1017 :
1018 PRINT "■■■■■■■■■■TRASPORTO SU";
1019 PRINT " O GIU' (S/G) ? ";
1020 GET UD$
1021 IF UD$ = "" THEN 1020
1022 :
1023 IF UD$ <> "S"  AND  UD$ <> "G" THEN 1020
1024 PRINT "■"; UD$; "■" :REM STAMPA
1025 :
1026 IF UD$ = "G" THEN ADJ = -ADJ
1027 FOR N = 1 TO 500 : NEXT N
1028 :
1029 :
2510 OCT = VAL(RIGHT$(NC$,1)) + ADJ
2513 IF OCT < 0 THEN OCT = OCT + 1 : GOTO 2513
2516 IF OCT > 7 THEN OCT = OCT - 1 : GOTO 2516
```

CAPITOLO NOVE

Effetti sonori speciali

In questo capitolo faremo in modo che il SID produca alcuni effetti sonori interessanti. Ascolteremo orologi, gong, un oscillatore del SID, cavalli, proiettili, misteriose entità pulsanti. Fra l'altro, parleremo di temporizzazione, inviluppo ADSR, modulazione ad anello, vibrato, spionaggio, collegamento, ritmo, rumore e variazioni di volume, frequenza e larghezza di impulso.
Tenete presente che la chiave per ottenere begli effetti sonori è la variazione fantasiosa: modifiche di volume, forma d'onda, frequenze, temporizzazione, ritmo e via dicendo. Ovviamente, bisogna sapere che cosa cambiare e come cambiarlo. In parte lo si può scoprire giocando con il SID e con programmi come quelli contenuti in questo capitolo. Dovrete anche passare un po' di tempo ad ascoltare il mondo che vi circonda: addestrate le vostre orecchie perché si perfezionino come strumenti di analisi dei suoni.

L'OROLOGIO

La figura 9.1 mostra il programma "Orologio". Leggetelo e fatelo girare. Giocate un po' con i numeri. Vedete se riuscite a ottenere un ritmo più interessante dal ticchettio dell'orologio.
Lavorando con gli effetti sonori si finisce per modificare molto i registri del SID, il che può dar luogo a qualche complicazione, se ci si dimentica su quali registri si è intervenuti. Tutti i programmi in questo capitolo cominciano e finiscono pulendo i registri del SID.

```
1000 REM *** OROLOGIO ***
1010 :
1020 :
1030 REM ** PULISCE IL SID E STAMPA IL PRONTO
1040 :
1050 SID = 54272        :REM CHIP SONORO
1060 FOR REG = SID TO SID+24
1070 :    POKE REG, 0
1080 NEXT REG
1090 :
1100 PRINT "♥";
1110 PRINT "LA BARRA SPAZIO PER FINIRE"
1120 :
1130 :
1140 REM ** INIZIALIZZA I REGISTRI DEL SID
1150 :
1160 POKE SID+6, 240   :REM SUSTAIN MAX
1170 POKE SID+24, 15   :REM VOLUME MAX
1180 :
1190 :
1200 REM ** SUONA; FINISCI ALLA PRESSIONE DI UN TASTO
1210 :
1220 POKE SID+1, 80              :REM TIC
1230 POKE SID+4, 17
1240 FOR T = 1 TO 3 : NEXT T
1250 POKE SID+4, 16
1260 FOR T = 1 TO 300 : NEXT T
1270 :
1280 POKE SID+1, 60              :REM TOC
1290 POKE SID+4, 17
1300 FOR T = 1 TO 3 : NEXT T
1310 POKE SID+4, 16
1320 FOR T = 1 TO 300 : NEXT T
1330 :
1340 GET KP$
1350 IF KP$ = "" THEN 1220
1360 :
1370 :
1380 REM ** PULISCE E FINISCE
1390 :
1400 FOR REG = SID TO SID+24
1410 :    POKE REG, 0
1420 NEXT REG
1430 PRINT "♥";
1440 :
1450 END
```

FIG. 9.1. Listato del programma "Orologio".

Consideriamo l'inviluppo ADSR che il programma genera. Velocità di attacco, decadimento e rilascio sono fissate a 0, e il livello di sostegno è 15, cioè il massimo. Il suono salirà rapidamente fino al livello massimo, decadrà rapidamente allo stesso livello (?), e vi rimarrà fino a che non verrà innescato il rilascio, per poi cadere velocemente a zero. La figura 9.2 rappresenta questo inviluppo.

Una volta regolati l'inviluppo e il volume generale, il programma è pronto per suonare una serie di tic-tac. Innanzitutto, le righe 1220-1260 eseguono il tic. La 1220 fissa una frequenza, poi la 1230 seleziona la

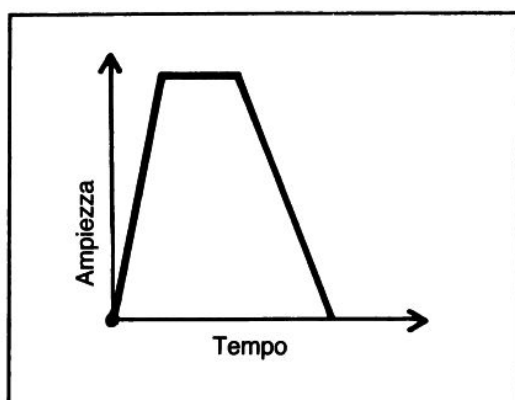



FIG. 9.2. Una raffigurazione dell'inviluppo ADSR usato nel programma "Orologio".

forma d'onda triangolare e avvia il suono. Segue una breve pausa, con il tic a volume massimo, poi la linea 1250 rilascia il suono. Infine, vi è una pausa relativamente lunga.
Poi le linee 1280-1320 danno il toc. Viene fissata una nuova frequenza, più bassa. Poi viene avviato il suono, tenuto per un po' e infine rilasciato. Di nuovo, si ha una pausa relativamente lunga. La linea 1340 esplora la tastiera: se non è stato premuto alcun tasto, il programma ritorna alla linea 1220 per un altro tic.
La parte in alto della figura 9.3 mostra le informazioni relative al volume

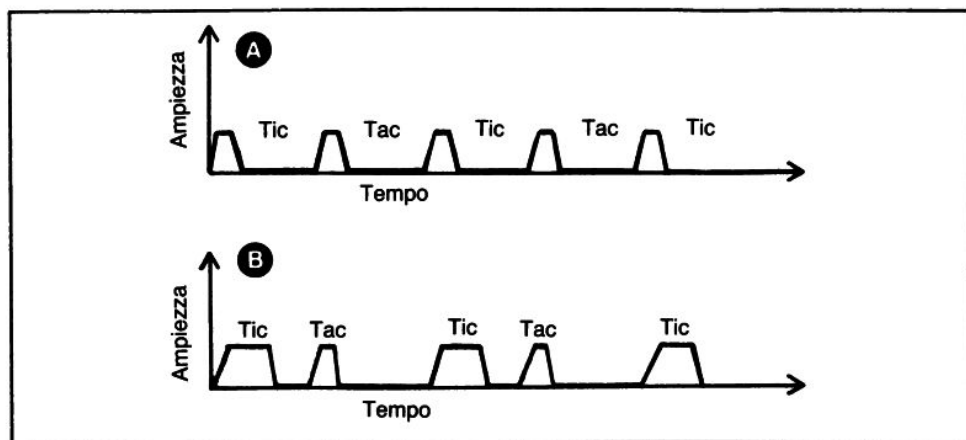



FIG. 9.3. Sopra: l'informazione del volume per alcuni movimenti del programma "Orologio" (non in scala). Sotto: una variazione possibile, con un ticchettio meno uniforme.

(non in scala) per questo programma, relativamente ad alcuni movimenti. Si noti la regolarità dell'andamento. La parte inferiore mostra che cosa accadrebbe se il tic avesse un sostegno più lungo e il toc seguisse più presto. Vedete se vi riesce di modificare l'orologio in modo che il suo ticchettio si avvicini all'andamento della parte inferiore della figura. La mia esperienza personale è che disegni grossolani come questo danno una prima idea di come si debbano regolare i registri del SID e di come si debbano organizzare i cicli di ritardo, quando si deve programmare un nuovo suono.
Quando si creano effetti sonori, servono programmi che possano essere modificati con facilità. Inserite molti cicli di ritardo e enunciati che regolano i registri del SID. Sono necessari sempre molti piccoli ritocchi prima di poter riprodurre i suoni che si sono immaginati.

LA MACCHINA DEL GONG

Abbiamo fatto produrre al SID qualche ticchettio d'orologio; facciamogli produrre ora qualche suono di gong, robusto e ricco di riverbero. Cominceremo con il prendere in considerazione la modulazione ad anello, che è un modo per collegare due voci.

Modulazione ad anello

Esiste una quinta possibilità, per la forma d'onda, sul SID, che ancora non ho citato: è la modulazione ad anello. Il SID può combinare l'informazione fornita da due voci per formare un'uscita modulata ad anello. Questo tipo di modulazione svolge una funzione eccellente per creare suoni come gong, campane, campanelli e simili.
Ecco come si procede per generare un'uscita modulata ad anello. In primo luogo, si sceglie la forma d'onda triangolare della voce. Poi, si regola il bit di controllo della modulazione ad anello, che è il bit 2 del registro di controllo della forma d'onda, e lo si porta a 1. Infine, si regola il partner della voce su una frequenza diversa da zero.
Che cos'è un partner? Quando si predispone una voce per la modulazione ad anello, questa miscela alla propria anche l'informazione sulla frequenza di un'altra voce. La voce 1 usa come partner la voce 3, la voce 2 usa la 1, la voce 3 usa la 2.
Un esempio. Predisponiamo la voce 1 per la modulazione ad anello. Si devono regolare i seguenti bit del registro di controllo dell'onda a SID + 4: bit 0 per innescare l'avvio di un inviluppo ADSR, bit 2 per

Valore del bit	128	64	32	16	8	4	2	1	
Numero del bit	7	6	5	4	3	2	1	0	
Funzione del bit	Rumore	Impulso	Dente di sega	Triangolare	—	Modulazione ad anello	Sincronizzazione	Porta	
	0	0	0	1	0	1	0	1	
				16	+	4	+	1	= 21

FIG. 9.4. Come si regola il quinto registro di una voce per la modulazione ad anello.

scegliere la modulazione ad anello, e bit 4 per selezionare la forma d'onda triangolare. Sommando i valori di questi bit si ottiene 21, e 21 è dunque il numero da porre in SID + 4. (Osservate la figura 9.4.) Poi si deve regolare la voce 3 su una frequenza diversa da zero, e lo si può fare assegnando al registro della frequenza in SID + 15 un valore non nullo, per esempio 19. Quando è il momento di innescare lo stadio di rilascio dell'inviluppo ADSR, basta inserire il valore 20 (bit 0 spento) in SID + 4.

Il programma

La figura 9.5 presenta il listato del programma "Macchina del gong", che usa la modulazione ad anello per creare nove diversi suoni di campanello. Leggetelo, poi battetelo, salvatelo e mandatelo in esecuzione.

Dopo che è stato pulito SID e lo schermo è stato inizializzato, le linee 1180-1190 regolano l'inviluppo ADSR per la voce 1. La linea 1210 fissa il volume generale.

Le linee 1260-1310 controllano i tasti premuti. Se si preme la barra spazio il programma ha termine. Se si preme uno dei tasti numerici 1-9 si genererà un suono di gong. Ogni altro input da tastiera viene ignorato.

La linea 1330 fissa la frequenza della voce 1, sulla base del numero del tasto premuto. La 1340 fa lo stesso per la voce 3; la 1350 poi innesca un suono modulato ad anello.

Per aggiungere enfasi al suono, le linee 1360-1410 fanno "ondeggiare" la voce 1. Questo tipo di effetto viene denominato vibrato o tremulo. Mentre il programma genera questo vibrato, non perde d'occhio la

```
1000 REM *** MACCHINA DEL GONG ***
1010 :
1020 :
1030 REM ** PULISCE IL SID E STAMPA IL PRONTO
1040 :
1050 SID = 54272          :REM CHIP SONORO
1060 FOR REG = SID TO SID+24
1070 :   POKE REG, 0
1080 NEXT REG
1090 :
1100 PRINT "⊐";
1110 PRINT "PREMI I TASTI 1-9 PER I GONG"
1120 PRINT
1130 PRINT "E LA BARRA SPAZIO PER FINIRE."
1140 :
1150 :
1160 REM ** INIZIALIZZA I REGISTRI DEL SID
1170 :
1180 POKE SID+5,12  :REM ATT=0, DEC=12
1190 POKE SID+6,9   :REM SOS=0, RIL=9
1200 :
1210 POKE SID+24,15 :REM VOLUME MASSIMO
1220 :
1230 :
1240 REM ** SUONA
1250 :
1260 GET KP$          :REM ESPLORA TASTIERA
1270 IF KP$ = "" THEN 1260
1280 IF KP$ = " " THEN 1480 :REM END IT
1290 :
1300 KP = VAL (KP$)   :REM DEVE ESSERE 1-9
1310 IF KP<1 OR KP>9 THEN 1260
1320 :
1330 POKE SID+1, KP * 1.5 + KP
1340 POKE SID+15, 19 + KP
1350 POKE SID+4, 21   :REM COLPO DI GONG
1360 FOR T = 1 TO 100
1370 :   QUAVER = T - INT(T/10)*10
1380 :   POKE SID, QUAVER * 20
1390 :   GET KP$
1400 :   IF KP$ <> "" THEN T = 100
1410 NEXT T
1420 POKE SID+4, 20   :REM IL SUONO CESSA
1430 GOTO 1270
1440 :
1450 :
1460 REM ** PULISCE TUTTO E FINISCE
1470 :
1480 FOR REG = SID TO SID+24
1490 :   POKE REG, 0
1500 NEXT REG
1510 PRINT "⊐";
1520 :
1530 END
```

FIG. 9.5. Listato del programma "Macchina del gong".

tastiera e, se viene premuto un tasto, il vibrato cessa, viene avviata la fase di rilascio dell'inviluppo e il programma ritorna a esplorare la tastiera. Se durante il vibrato non viene premuto alcun tasto, il gong svanisce lentamente, e il programma ritorna a esplorare la tastiera.

Ho passato parecchio tempo a provare l'effetto di diverse formule per le linee 1330 e 1340. La relazione fra le frequenze delle due voci e il suono risultante, modulato ad anello, è complessa. Potete provare a escogitare qualche altra formula di testa vostra.
Un altro punto dove val la pena di esercitare un po' di sperimentazione è la linea 1370, la formula del vibrato. Modificando questa linea si possono ottenere interessanti variazioni sul tema del gong.

IL SID SI ASCOLTA

La modulazione ad anello fa sì che una voce ne influenzi un'altra: è già una possibilità di controllo, ma in talune situazioni si vorrebbe qualcosa di più. Sarebbe bello se si potesse "origliare" e spiare qualcosa dell'uscita del SID. I registri in SID + 27 e SID + 28 consentono proprio questo e offrono un modo più controllato per collegare insieme più voci.

I registri "spia"

SID + 27 presenta l'uscita dell'oscillatore della voce 3. SID + 28 presenta l'uscita del generatore di inviluppo della voce 3. Si possono leggere questi registri, per poi usarne i valori per modificare altre regolazioni del SID.
Dovete mettere in funzione l'oscillatore della voce 3 perché SID + 27 mostri variazioni di valore. Per questo stabilite una frequenza e una forma d'onda per la voce 3. Non sentirete la voce 3 finché non innescherete l'inviluppo ADSR: pertanto la voce 3 può continuare a oscillare, senza produrre alcun suono, mentre se ne leggono le oscillazioni da SID + 27.
Dovete innescare il generatore di inviluppo della voce 3 perché i suoi valori siano leggibili in SID + 28. Di solito questo farà sì che la voce 3 emetta qualche suono. Per evitarlo, ma avere egualmente la possibilità di controllare il suo generatore di inviluppo, zittite la voce regolando il bit 7 di SID + 24 sul valore 1. SID + 24 è lo stesso registro usato per regolare il volume generale. Per portare il bit 7 a 1 è sufficiente sommare 128 al valore del volume e inserire con una istruzione POKE il nuovo valore.

Il calcolatore pazzo

Consideriamo ora un programma che sfrutta queste possibilità di spiare il SID: il programma “Computer pazzo”, il cui listato è dato in figura 9.6. Leggete, battetelo, salvatelo e mandatelo in esecuzione. Premendo

```
1000 REM *** COMPUTER PAZZO ***
1010 :
1020 :
1030 REM ** PULISCE IL SID E STAMPA IL PRONTO
1040 :
1050 SID = 54272
1060 FOR N = SID TO SID+24
1070 :   POKE N,0
1080 NEXT N
1090 :
1100 PRINT "♥";
1110 PRINT "PREMI I TASTI 1-9 PER CAMBIARE"
1120 PRINT
1130 PRINT "E QUALSIASI ALTRO TASTO PER FINIRE"
1140 :
1150 :
1160 REM ** INIZIALIZZA I REGISTRI DEL SID
1170 :
1180 POKE SID+6,240   :REM SUST VOCE 1 = MAX
1190 :
1200 POKE SID+15,18   :REM FISSA FREQ VOCE 3
1210 POKE SID+18,16   :REM FISSA FORMA D'ONDA VOCE 3
1220 :
1230 POKE SID+24,15   :REM REGOLA VOLUME
1240 :
1250 :
1260 REM ** SUONA
1270 :
1280 POKE SID+4,17    :REM INNESCA ATT VOCE 1
1290 :  :REM FISSA FREQ VOCE 1 IN BASE ALLE
1291 :  :REM OSCILLAZIONI DELLA VOCE 3
1300 POKE SID+1, PEEK(SID+27)
1310 FOR T = 1 TO 5   :REM ASPETTA UN PO'
1320 NEXT T
1330 :
1340 :
1350 REM ** ESPLORA LA TASTIERA PER STABILIRE SE
1351 REM    SUONARE ANCORA, CAMBIAR SUONO O FINIRE
1360 :
1370 GET KP$
1380 IF KP$ = "" THEN 1300 :REM ANCORA
1390 :
1400 IF ASC(KP$)<49 THEN 1450
1410 IF ASC(KP$)>58 THEN 1450
1420 :  POKE SID+15, VAL(KP$) * 7
1430 :  GOTO 1300      :REM SUONO CAMBIATO
1440 :
1450 FOR REG = SID TO SID+24 :REM PULISCE
1460 :   POKE REG, 0          :REM TUTTO E
1470 NEXT REG                 :REM FINISCE
1480 PRINT "♥";
1490 :
1500 END
```

FIG. 9.6. Listato del programma “Computer pazzo”.

uno qualunque dei tasti numerici 1-9, l'andamento del suono cambierà; la pressione di qualunque altro tasto farà cessare il programma.
In questo programma, la voce 1 produce suoni le cui frequenze sono basate sulle oscillazioni della voce 3.
La chiave di volta è la linea 1300, che prende un valore da SID + 27 e lo inserisce in uno dei registri di frequenza della voce 1. Dopo una breve pausa, il programma controlla se viene premuto un tasto.
Quali valori si presenteranno in SID + 27? Bisogna considerare come oscilla la voce 3. Poiché la linea 1210 seleziona la forma d'onda triangolare, l'uscita della voce 3 andrà da 0 a 255 e viceversa, a una velocità determinata dalla sua frequenza. I valori raccolti alla linea 1300 dipenderanno da questa frequenza e dalla frequenza del campionamento.
Ora, in prevalenza la voce 1 campiona SID + 27 a frequenza costante, interrompendosi solo qualora venga premuto un tasto sulla tastiera. I campioni che raccoglierà, e quindi i suoni che produrrà, presenteranno un certo andamento. Premendo uno dei tasti numerici, la frequenza della voce 3 cambia e la voce 1, che ancora sta osservando le oscillazioni della voce 1 a frequenza costante, comincerà a vedere schemi di dati diversi, e anche l'andamento del suono muterà.
Un ultimo fatto interessante a proposito del programma: il volume della voce 1 sale al livello massimo e vi rimane fino a che il programma termina. Questo risultato si ottiene con due regolazioni. In primo luogo, viene fissato al massimo il livello di sostegno.
In secondo luogo, lo stadio di rilascio dell'inviluppo ADSR non viene innescato finché il programma non termina. La figura 9.7 mostra quale aspetto abbia un simile inviluppo.

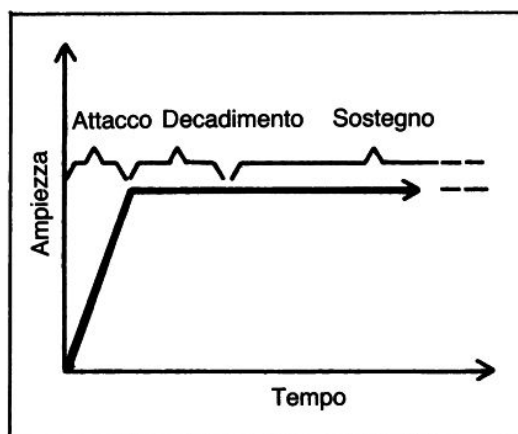



FIG. 9.7. Una rappresentazione dell'inviluppo ADSR usato nel programma "Computer pazzo".

Tatatum tatatum, tatatum tum tum

Il prossimo programma usa vari cicli di temporizzazione per simulare il suono di un cavallo al galoppo. (Per ispirarvi, ripensate al cavallo dell'uomo mascherato delle vecchie storie che avrete sicuramente ascoltato da ragazzi.)
La figura 9.8 dà il listato del programma "Cavallo". Dopo che lo avrete

```
1000 REM *** CAVALLO ***
1010 :
1020 :
1030 REM ** PULISCE IL SID E STAMPA IL PRONTO
1040 :
1050 SID = 54272          :REM CHIP SONORO
1060 FOR REG = SID TO SID+24
1070 :    POKE REG, 0
1080 NEXT REG
1090 :
1100 PRINT "J";
1110 PRINT "PREMI LA BARRA SPAZIO PER FINIRE."
1120 :
1130 :
1140 REM ** INIZIALIZZA I REGISTRI DEL SID
1150 :
1160 POKE SID+5, 4    :REM ATT=0, DEC=4
1170 POKE SID+6, 164 :REM SOS=10, RIL=4
1180 :
1190 :
1200 REM ** REGOLA VOLUME, FREQUENZA, TEMPO
1210 :
1220 VC = 1           :REM CAMBIA VOLUME
1230 VLM = 12         :REM INIZIA VOLUME
1240 :
1250 VLM = VLM + VC   :REM AGGIORNA VOLUME
1260 IF VLM = 15 OR VLM = 12 THEN VC = -VC
1270 POKE SID+24, VLM
1280 :
1290 FRQ = 35 - VLM   :REM RELAZIONE FRA FREQ E VOL
1300 DLY = 17         :REM FATTORE TEMPO
1310 :
1320 :
1330 REM ** I QUATTRO ZOCCOLI
1340 :
1350 POKE SID+1, FRQ + 2  :REM ZOCCOLO 1
1360 POKE SID+4,129
1370 FOR T = 1 TO DLY*1.1 : NEXT T
1380 POKE SID+4, 128
1390 FOR T = 1 TO DLY * 3 : NEXT T
1400 :
1410 POKE SID+1, FRQ      :REM ZOCCOLO 2
1420 POKE SID+4,129
1430 FOR T = 1 TO DLY : NEXT T
1440 POKE SID+4, 128
1450 FOR T = 1 TO  DLY * 1.1 : NEXT T
1460 :
1470 POKE SID+1, FRQ - 2  :REM ZOCCOLO 3
1480 POKE SID+4,129
1490 FOR T = 1 TO DLY * 1.2: NEXT T
1500 POKE SID+4, 128
```

```
1510 FOR T = 1 TO DLY * 1.4 :NEXT T
1520 :
1530 POKE SID+1, FRQ      :REM ZOCCOLO 4
1540 POKE SID+4,129
1550 FOR T = 1 TO DLY * .8: NEXT T
1560 POKE SID+4, 128
1570 FOR T = 1 TO DLY * 5.5 : NEXT T
1580 :
1590 :
1600 REM ** SI FERMA ALLA PRESSIONE DI UN TASTO
1610 :
1620 GET KP$
1630 IF KP$ = "" THEN 1250
1640 :
1650 FOR REG = SID TO SID+24
1660 :    POKE REG, 0
1670 NEXT REG
1680 PRINT "♥";
1690 :
1700 END
```

FIG. 9.8. Listato del programma "Cavallo".

fatto girare, modificate i ritmi divertendovi a variare le formule della temporizzazione. Riuscite a far andare il cavallo al galoppo? A farlo impennare? A farlo procedere al passo per tutta la strada? Tutto sta nella temporizzazione.
Esaminiamo il programma. Il primo segmento realizza come al solito la pulizia del SID e la visualizzazione del "pronto". Il segmento successivo fissa l'inviluppo ADSR per lo scalpitare degli zoccoli. Questo suono ha un inviluppo molto classico. Sale rapidamente al volume massimo, scende a velocità moderata, si ferma a circa due terzi del volume massimo, e infine si estingue a velocità moderata. Potete dare l'impressione di vari tipi di cavalli, di zoccoli e di superfici, modificando le regolazioni dell'inviluppo e della forma d'onda.
Le linee 1220-1290 costituiscono un segmento interessante. Ogni volta che il programma passa per queste istruzioni, le regolazioni di volume e di frequenza verranno leggermente modificate. Queste variazioni faranno sì che il suono degli zoccoli risulterà un po' più naturale. La linea 1300 fissa una variabile fondamentale di temporizzazione: il resto della temporizzazione si baserà sul valore di DLY. Si può provare anche a inserire una formula che faccia variare ogni tanto il valore di DLY.
Le linee 1350-1570 generano il rumore degli zoccoli, uno alla volta. Per ogni zoccolo, viene attivata la voce 1; segue un breve ritardo; la voce viene rilasciata; segue poi un ritardo maggiore. I ritardi variano da zoccolo a zoccolo; come per i fiocchi di neve, non esistono due zampe uguali.

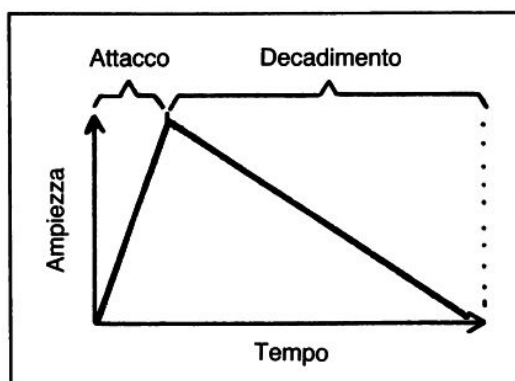



FIG. 9.9. Una rappresentazione dell'inviluppo ADSR che dovreste provare a programmare per simulare un colpo di pistola.

Vedete se vi riesce di far sembrare che il cavallo si avvicini lentamente, vi passi accanto e se ne vada. Ecco tre utili suggerimenti:
– Quando un suono si avvicina, diventa progressivamente più forte, e la sua frequenza cresce.
– Quando un suono si allontana, diventa progressivamente più fievole e la sua frequenza diminuisce.
– Un po' di esagerazione non guasta mai, negli effetti sonori.

BANG BANG

Proviamo a pensare come si potrebbe simulare il suono di una pistola. I suoni a cui pensare sono in realtà due. In primo luogo, una piccola esplosione, quando la polvere si innesca e viene lanciato il proiettile. Poi c'è il suono della pallottola che fende l'aria.
Per le esplosioni torna comodissimo il rumore bianco: il bit 7 del registro della forma d'onda di ciascuna voce permette di selezionare il rumore bianco, e potremo iniziare ogni colpo di pistola con uno scoppio di rumore bianco. Le esplosioni, poi, iniziano con un rumore forte, e poi progressivamente svaniscono. Dovrete cercare di fissare un inviluppo ADSR dall'aspetto simile a quello di figura 9.9.
Passiamo ora al fischio della pallottola che viaggia nell'aria. Ci vuole un po', dopo l'esplosione, perché la pallottola assuma una velocità sufficiente a farcela udire. Quando accelera verso un ascoltatore, il suo

suono diventa più acuto e più forte. Quando passa e si allontana dall'ascoltatore, il suono diminuisce di altezza e di volume. Servirà un inviluppo ADSR che produca una discesa e una risalita di volume, poi dovremo stabilire qualche ciclo di regolazione della frequenza che vada di pari passo con le variazioni di volume.

Come si generano i suoni

La figura 9.10 dà il listato del programma "BamP'Twang", che genera rumori di spari. Fatelo girare. Che cosa ve ne sembra? Vorrete aggiungere un'eco con una terza voce, oppure vorrete mettere a punto la temporizzazione, o ancora vorrete modificare le frequenze. Come al solito, un po' di sperimentazione vi insegnerà moltissimo.
Le linee 1180-1220 fissano due inviluppi ADSR. La voce 1 tratterà l'esplosione, e la voce 2 il percorso della pallottola nell'aria. La voce 1, con una velocità di attacco 0, farà sì che si raggiunga il volume massimo in 2 millesimi di secondo, dopodiché comincerà il decadimento, a una velocità molto più lenta, di 1,5 secondi. La voce 2 ha una velocità di attacco pari a 5: ci vorranno 55 millesimi di secondo perché venga raggiunto il volume massimo, poi il suono decadrà a una velocità prossima a quella della voce 1. Fate girare il programma con valori diversi per la definizione degli inviluppi ADSR: potrete simulare tipi diversi di pistole e di pallottole.
Dopo di questo il programma aspetta, alle righe 1290-1300, che venga premuto un tasto. Se si preme il tasto RETURN il programma termina, mentre ogni altro tasto fa sì che venga sparata una pallottola. Lo sparo vero e proprio avviene alle linee 1360-1400.
Prima viene la voce 1, con l'esplosione. Notate che l'esplosione precedente non viene rilasciata completamente fino all'ultimo momento. Segue una breve pausa (linea 1380), così che la pallottola possa assumere un po' di velocità. Poi entra in gioco la voce 2, con il fischio della pallottola in movimento.
Le linee 1420-1480, poi, fanno variare regolarmente la frequenza della voce 2. A differenza di "Macchina del gong", questo programma non esplora la tastiera mentre sta giocando con le frequenze. Questo significa che non si può sparare molto velocemente. Provate a ovviare a questa limitazione.
Il rumore bianco è eccellente anche per la simulazione di onde, vento, porte che sbattono, e fenomeni analoghi. È particolarmente interessante combinarlo con forme d'onda più musicali, come fa quest'ultimo programma.

```
1000 REM *** BAM-P'TWANG ***
1010 :
1020 :
1030 REM ** PULISCE IL SID E STAMPA IL PRONTO
1040 :
1050 SID = 54272       :REM CHIP SONORO
1060 FOR REG = SID TO SID+24
1070 :    POKE REG, 0
1080 NEXT REG
1090 :
1100 PRINT "";
1110 PRINT "PREMI LA BARRA SPAZIO PER SPARARE"
1120 PRINT
1130 PRINT "E IL TASTO RETURN PER FINIRE."
1140 :
1150 :
1160 REM ** INIZIALIZZA I REGISTRI DEL SID
1170 :
1180 POKE SID+5,10      :REM V-1 ATT/DEC
1190 POKE SID+1, 10     :REM V-1 FREQ
1200 :
1210 POKE SID+12,89     :REM V-2 ATT/DEC
1220 POKE SID+13,10     :REM V-2 SUS/RIL
1230 :
1240 POKE SID+24, 15    :REM VOLUME MAX
1250 :
1260 :
1270 REM ** ESPLORA LA TASTIERA PER SPARARE O FINIRE
1280 :
1290 GET KP$
1300 IF KP$ = "" THEN 1290
1310 IF KP$ = CHR$(13) THEN 1550
1320 :
1330 :
1340 REM ** ESEGUE  : VOCE 1 ESPLOSIONE,
1341 REM              VOCE 2 VOLO
1350 :
1360 POKE SID+4,128    :REM RILASCIO V-1
1370 POKE SID+4,129    :REM INIZIO V-1
1380 REM  FOR T = 1 TO 20: NEXT T
1390 POKE SID+11, 16   :REM RILASCIO V-2
1400 POKE SID+11, 17   :REM INIZIO V-2
1410 :
1420 FOR FRQ = 10 TO 80 STEP 3
1430 :    POKE SID+8, FRQ
1440 NEXT FRQ
1450 FOR FRQ = 77 TO 5  STEP -3
1460 :    POKE SID+8, FRQ
1470 :    FOR T = 1 TO 4 : NEXT T
1480 NEXT FRQ
1490 :
1500 GOTO 1290
1510 :
1520 :
1530 REM ** PULISCE TUTTO E FINISCE
1540 :
1550 FOR REG = SID TO SID+24
1560 :    POKE REG, 0
1570 NEXT REG
1580 PRINT "";
1590 :
1600 END
```

FIG. 9.10. Listato del programma "Bam-P'Twang".

LA ZONA PULSANTE

L'ultimo effetto sonoro combina forme d'onda a impulso di varia larghezza con variazioni di volume regolari.
Questo crea un rumore misterioso che sarà perfetto per i raggi disintegratori o per la musica di sottofondo per un'atmosfera crepuscolare.
Il listato del programma è riportato in figura 9.11: leggetelo come al

```
1000 REM *** ZONA PULSANTE ***
1010 :
1020 :
1030 REM ** PULISCE IL SID E STAMPA IL PRONTO
1040 :
1050 SID = 54272          :REM SOUND CHIP
1060 FOR REG = SID TO SID+24
1070 :    POKE REG, 0
1080 NEXT REG
1090 :
1100 PRINT "⌐";
1110 PRINT "PREMI LA BARRA SPAZIO PER FINIRE"
1120 :
1130 :
1140 REM ** INIZIALIZZA I REGISTRI DEL SID
1150 :
1160 POKE SID+1, 20     :REM V-1 FREQ
1170 POKE SID+6, 240    :REM V-1 SOS/RIL
1180 :
1190 POKE SID+24, 15    :REM VOLUME MAX
1200 :
1210 :
1220 REM ** PLAY IT
1230 :
1240 POKE SID+4, 65   :REM V-1 IMPULSO ATTIVATO
1250 :
1260 VLM = 6 : A = -3
1270 IF VLM = 15 OR VLM = 6 THEN A = -A
1280 VLM = VLM + A
1290 POKE SID+24, VLM   :REM REGOLA VOL
1300 :
1310 FOR N = 8 TO 15    :REM AMPIEZZA IMPULSO
1320 :    POKE SID+3, N :REM IN AUMENTO
1330 NEXT N
1340 :
1350 FOR N = 14 TO 9 STEP -1 :REM AMPIEZZA
1360 :    POKE SID+3, N       :REM IMPULSO IN
1370 NEXT N                   :REM DIMINUZIONE
1380 :
1390 :
1400 REM ** ESPLORA LA TASTIERA
1410 :
1420 GET KP$
1430 IF KP$ = "" THEN 1270   :REM NESSUN TASTO
1440 :
1450 :
1460 REM ** PULISCE TUTTO E FINISCE
1470 :
```

```
1480 FOR REG = SID TO SID+24
1490 :    POKE REG, 0
1500 NEXT REG
1510 PRINT "";
1520 :
1530 END
```

FIG. 9.11. Listato del programma "Zona pulsante".

solito, battetelo e memorizzatelo; fatelo girare, apportate le modifiche di vostro gusto.
Le linee 1160-1190 fissano frequenza, inviluppo ADSR e volume. Come in "Calcolatore pazzo", il volume sale rapidamente al massimo e vi rimane fino a che il programma termina.
La linea 1240 seleziona la forma d'onda a impulso per la voce 1 e innesca l'inviluppo ADSR. La linea 1260 dà i valori iniziali per il volume e le variabili di variazione del volume.
La linea 1270 è l'inizio del ciclo principale di programma. Il volume generale oscillerà fra i valori 6 e 15. La linea 1270 commuta la direzione dei cambiamenti di volume, ogniqualvolta vengono raggiunti i valori limite. La linea 1280 modifica il volume sommando la variazione, poi la linea 1290 inserisce con l'istruzione POKE il nuovo valore.
Le linee 1310-1350 fanno variare la regolazione della larghezza dell'impulso da 8 a 15, un'unità alla volta.
Si ottengono così larghezze di impulso dal 50 al 94 per cento. (Abbiamo visto come si regola la larghezza d'impulso nel capitolo 7.)
Le linee 1350-1370 poi riducono la larghezza dell'impulso di una unità alla volta. Le 1420-1430 esplorano rapidamente la tastiera: se è stato premuto un tasto, il programma termina; altrimenti si torna alla linea 1270 per una nuova regolazione di volume e un altro passaggio nei cicli di variazione della larghezza d'impulso.
Fra le variazioni e le aggiunte che potete apportare al programma vi sono variazioni di frequenza, modulazione ad anello, effetti di eco, una seconda voce con larghezze di impulso che variano seguendo un andamento contrario, e un diverso inviluppo ADSR. Come sempre, sperimentare con un po' di fantasia vi permetterà di imparare molte cose.

CHE COSA ABBIAMO IMPARATO

In questo capitolo abbiamo visto sei diversi programmi per effetti sonori. Ecco gli argomenti che abbiamo affrontato:

– Come usare brevi impulsi di forme d'onda triangolari per simulare il ticchettio di un orologio
– Come usare la modulazione ad anello e variazioni di frequenza per simulare i gong
– Come usare le informazioni dell'oscillatore della voce 3 per modulare la frequenza di un'altra voce, simulando così un calcolatore pazzo
– Come usare vari cicli di temporizzazione per simulare i suoni ritmici di un cavallo al galoppo
– Come miscelare rumore bianco e onda triangolare per simulare lo sparo di una pistola
– Come variare larghezza di impulso e volume per creare un suono misterioso, da film dell'orrore.

Negli ultimi tre capitoli abbiamo dato uno sguardo alle capacità di produzione di suoni del SID: nel capitolo 10 uniremo le forze del SID e del VIC in programmi che combinano suono e grafica.

ESERCIZI

Test

1. Si può rallentare il ticchettio, nel programma "Orologio", usando numeri .... nei cicli di ritardo delle linee 1260 e 1320.
2. Per produrre la modulazione ad anello si usano .... voci.
3. I registri SID + 27 e SID + 28 permettono di spiare le attività della ....
4. Nel programma "Cavallo" si usano piccole variazioni di volume e di frequenza per rendere il suono più ....
5. Il programma "Bam-P'Twang" usa il .... per simulare l'esplosione della polvere da sparo.
6. I cicli alle linee 1310-1370 del programma "Zona pulsante" sono usati per modificare .... della voce 1.

Esercizi di programmazione

1. Modificate il programma "Orologio" in modo che sfrutti tutte e tre le voci, creando un suono più ricco.
2. Modificate il programma "Bam-P'Twang" in modo che il suono esplosivo arrivi dopo il volo della pallottola in aria.

3. Modificate il programma "Zona pulsante" facendo in modo che la frequenza della voce 1 cambi con la sua larghezza di impulso.

Risposte al test

1. più grandi
2. due
3. voce 3
4. naturale
5. rumore o rumore bianco
6. la larghezza d'impulso

Soluzioni possibili agli esercizi di programmazione

1. Caricate il programma "Orologio", poi battete queste linee:

```
1000 REM *** OROLOGIO RICCO ***
1163 POKE SID+13, 120 : REM V-2 SOS/RIL
1166 POKE SID+20, 180 : REM V-E SOS/RIL
1223 POKE SID+8, 20
1226 POKE SID+15, 40
1233 POKE SID+11, 17
1236 POKE SID+18, 17
1253 POKE SID+11, 16
1256 POKE SID+18, 16
1283 POKE SID+8,  15
1286 POKE SID+15, 30
1293 POKE SID+11, 17
1296 POKE SID+18, 17
1313 POKE SID+11, 16
1316 POKE SOD+18, 16
```

2. Caricate il programma "Bam-P'Twang", poi battete queste linee:

```
1000 REM *** P'TWANG-BAM ***
1360 :
1370 :
1380 :
1492 POKE SID+4, 128     :REM RIL V-1
1494 POKE SID+4, 129     :REM INIZIO V-1
1496 REM  FOR T = 1 TO 20: NEXT T
1498 :
```

3. Caricate il programma "Zona pulsante", poi battete queste linee:

```
1000 REM *** ALTRA ZONA PULSANTE ***
1160 :
1325 :    POKE SID+1, 2 * N :REM V-1 FRQ
1365 :    POKE SID+1, 2 * N :REM V-1 FRQ
```

Suoni + grafica = magia

Nei primi sei capitoli, siamo andati alla scoperta di alcune fra le capacità grafiche del Commodore 64. Negli ultimi tre capitoli, poi, abbiamo imparato come produrre suoni. Ora è venuto il momento di unire grafica e suoni. Vi presenterò tre programmi a questo proposito e, lungo la strada, discuterò alcune fra le tecniche di progettazione che ho trovato utili per questo tipo di programmazione.

SINERGIA

"Sinergia" è una parola che deriva dalla biologia, e descrive situazioni in cui due o più cose si uniscono e provocano effetti molto al di là di quelli che ciascuna componente potrebbe produrre da sola. Si può esprimere la cosa in un altro modo: l'intero diventa maggiore della somma delle sue parti.
L'unione di immagini e suoni in modo abile può creare effetti splendidi. Provate a immaginare Guerre stellari senza la sua colonna sonora, oppure pensate a che cosa sarebbe giocare a Donkey Kong in versione senza suoni.
Buoni effetti sonori rendono più facile memorizzare immagini e buone immagini favoriscono l'idea di alcuni suoni. Se i due elementi sono riuniti con cura, esercitano un'azione sinergica, creando un nuovo livello di illusione.
I migliori programmatori passano molto tempo a perfezionare i loro effetti sonori e grafici. Questa fase può essere frustrante, se si lavora con

un programma mal progettato. Invece mettere a punto fin nei dettagli un programma ben progettato può essere davvero molto divertente. Che cosa fa di un programma un programma ben progettato? Uno dei fattori più importanti è la modularità.

PENSARE PER MODULI

Per un programmatore alle prime armi è facile imparare le regole di un linguaggio di programmazione e le caratteristiche di un calcolatore particolare, ma la parte difficile è imparare come si mette insieme un programma di grandi dimensioni.
I buoni programmatori cominciano con il pensare. Prendono un problema complesso e cominciano a suddividerlo in pezzi più semplici, dei moduli. Poi suddividono i moduli complessi in pezzi ancora più semplici e procedono in questo modo fino a che non ottengono un insieme di moduli semplici che coprono ogni dettaglio del problema di partenza. Allora cominciano a tradurre il loro progetto in specifiche istruzioni di calcolatore.
Questa impostazione viene chiamata programmazione strutturata di tipo "top-down", cioè "dall'alto verso il basso", e la si può usare con qualunque linguaggio di programmazione e qualunque calcolatore. Per i principianti può sembrare una pura perdita di tempo: vogliono sedersi al calcolatore e cominciare a scrivere, ma basta qualche esperienza di lotta con un programma malstrutturato per intravvedere la luce.
Come si impara a programmare in questo modo? Si comincia con il leggere libri e riviste, con il parlare con altri programmatori, con l'esaminare programmi di ogni genere, con l'imparare più di un linguaggio di programmazione e con il tentare di prestare attenzione ai propri errori. Mantenete la vostra mente aperta, attenta e tranquilla – e scrivete tanti programmi.

BLIP E BIP

Circa 10 anni fa fece la comparsa il primo videogioco da casa di grande diffusione: Pong. I giocatori dovevano far rimbalzare un puntino luminoso da una parte all'altra dello schermo. Quando il puntino colpiva una parete o una racchetta simulata, si sentiva un piccolo bip. Il primo programma di questo capitolo è un piccolo omaggio al mondo semplice dei blip e dei bip.

```
1000 REM *** RIMBALZO ***
1010 :
1020 :
1030 REM ** DISEGNA IL RETTANGOLO E STAMPA IL PRONTO
1040 :
1050 BX$(1) = "╭──────────────────╮"
1060 BX$(2) = "│                  │"
1070 BX$(3) = "╰──────────────────╯"
1080 :
1090 PRINT "▯■■"          :REM PULISCE E GIU'
1100 PRINT SPC(10); BX$(1)   :REM CIMA
1110 FOR N = 1 TO 3          :REM LATI
1120 :    PRINT SPC(10); BX$(2)
1130 NEXT N
1140 PRINT SPC(10); BX$(3)   :REM FONDO
1150 :
1160 PRINT SPC(10); "■PREMI  UN ";
1170 PRINT "TASTO PER FINIRE"
1180 :
1190 :
1200 REM ** FISSA I DATI DEGLI SPRITE
1210 :
1220 FOR N = 12288 TO 12350 :REM PER LO PIU'
1230 :    POKE N, 0          :REM VUOTI
1240 NEXT N
1250 :
1260 FOR N = 12288 TO 12300 STEP 3
1270 : READ SPDTA
1280 : POKE N, SPDTA     :REM FORMA DELLA PALLINA
1290 NEXT N
1300 :
1310 DATA 60, 126, 255, 126, 60
1320 :
1330 :
1340 REM ** REGOLA I REGISTRI DEL VIC
1350 :
1360 VIC = 53248       :REM CHIP GRAFICO
1370 POKE 2040, 192   :REM PUNTA AI DATI
1380 POKE VIC+39, 7   :REM SPR 0 GIALLO
1390 POKE VIC+21, 1   :REM SPR 0 ACCESO
1400 :
1410 :
1420 REM ** INIZIALIZZA I SUONI
1430 :
1440 SID = 54272       :REM CHIP SONORO
1450 POKE SID+5, 24   :REM ATT=1, DEC=8
1460 POKE SID+24, 15 :REM VOLUME MAX
1470 :
1480 :
1490 REM ** INIZIALIZZA LA POSIZIONE
1491 REM  · DELLA PALLINA E LA MUOVE
1500 :
1510 HP = 180 : VP = 89  :REM POSITIONA
1520 HM = 4   : VM = 2.5 :REM MUOVE
1530 :
1540 :
1550 REM ** MUOVE LA PALLINA
1560 :
1570 HP = HP + HM     :REM NUOVA POS OR
1580 VP = VP + VM     :REM NUOVA POS VER
1590 POKE VIC, HP     :REM FISSA LE NUOVE
1600 POKE VIC+1, VP   :REM POSIZIONI
1610 :
1620 :
```

```
1630 REM ** CONTROLLA SE E' STATO PREMUTO UN TASTO
1640 :
1650 GET KP$
1660 IF KP$ <> "" THEN 1950 :REM FINISCE
1670 :
1680 :
1690 REM ** CONTROLLA SE C'E' STATO URTO
1700 :
1710 HH = (HP < 111  OR  HP > 249)
1720 VH = (VP < 80   OR  VP > 102)
1730 :
1740 IF (NOT HH) AND (NOT VH) THEN 1570
1750 :
1760 :
1770 REM ** AFFRONTA LA SITUAZIONE
1780 :
1790 IF HH THEN HM = -HM :REM GIRA
1800 IF VH THEN VM = -VM :REM GIRA
1810 :
1820 POKE SID+4, 16  :REM RILASCIA IL SUONO
1830 POKE SID+1, RND(0)*40 + 10
1840 POKE SID+4, 17  :REM INIZIO SUONO
1850 :
1860 HUE = (PEEK(VIC+39) AND 15) + 1
1870 IF HUE = 16 THEN HUE = 1
1880 POKE VIC+39,HUE  :REM CAMBIA COLORE
1890 :
1900 GOTO 1570       :REM RISOLTO L'URTO
1910 :
1920 :
1930 REM ** PULISCE TUTTO E TORNA A POSTO
1940 :
1950 POKE SID+24,0 :REM SUONO DISATTIVATO
1960 POKE VIC+21,0 :REM SPRITE DISATTIVATO
1970 PRINT "♥";    :REM PULISCE SCHERMO
1980 :
1990 END
```

FIG. 10.1. Listato del programma "Rimbalzo".

La figura 10.1 presenta il listato del programma "Rimbalzo". Battetelo, memorizzatelo, poi mandatelo in esecuzione.

Nella maggior parte delle visualizzazioni grafiche, vi sono parti dell'immagine che stanno ferme e parti che si muovono: possiamo definirle, rispettivamente, elementi statici ed elementi dinamici.

In "Rimbalzo" il riquadro è l'elemento statico, e il blip in movimento è l'elemento dinamico. Il riquadro è disegnato con caratteri grafici, e il blip è uno sprite. Con il Commodore 64, per gli elementi statici funzionano molto bene la mappa di bit e i caratteri grafici. Per gli elementi dinamici vanno bene i caratteri grafici e gli sprite.

Grafica e suono

Guardiamo i moduli di "Rimbalzo". Le linee 1050-1170 impostano gli elementi statici della visualizzazione. Sono usati caratteri di controllo del cursore, stringhe costituite da caratteri grafici e il comando SPC( ). I due moduli successivi impostano lo sprite. Le linee 1220-1310 caricano i dati per uno sprite molto semplice, visibile nella figura 10.2. Poi le linee 1360-1390 regolano i registri del VIC necessari. Le linee 1440-1460 inizializzano il chip della generazione di suoni. Il programma usa la voce 1. La linea 1450 fissa i valori per l'attacco e il decadimento della voce. La linea 1460 regola il livello del volume generale del SID. La frequenza e la forma d'onda per la voce 1 verranno regolate ogniqualvolta il blip va a urtare una parete.

Come si fa muovere il blip

La parte principale del programma forma un grande ciclo. Ogni volta che passa per il ciclo, il blip si sposta sullo schermo. Il movimento del

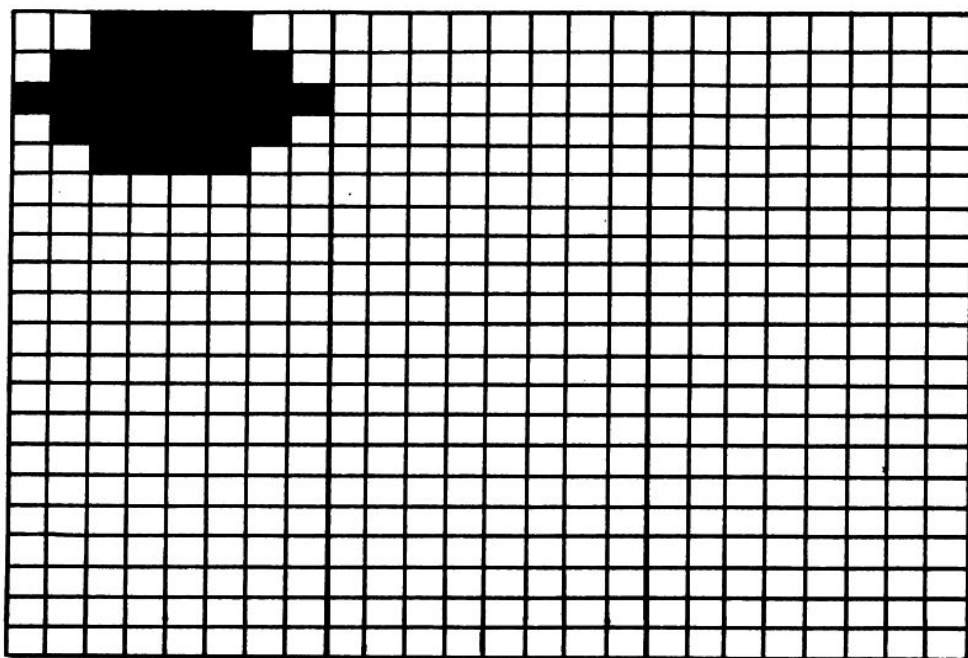



FIG. 10.2. Il semplice disegno di sprite usato nel programma "Rimbalzo".

blip è gestito da quattro variabili: HP e VP tengono conto della sua posizione orizzontale e verticale rispettivamente. HM contiene il valore e la direzione delle mosse orizzontali; VM contiene il valore e la direzione delle mosse verticali.
Le linee 1510-1520 inizializzano queste quattro variabili. Lo sprite viene posto al centro del riquadro disegnato alle linee 1050-1140, pronto a muoversi in orizzontale a velocità quasi doppia che in verticale.
La linea 1570 è l'inizio del ciclo principale del programma. Le linee 1570-1600 calcolano nuove posizioni orizzontali e verticali per il blip, poi le inseriscono nei registri di posizione dello sprite 0. Poi il programma controlla se è stato premuto un tasto: nel caso, si ha un salto al modulo di chiusura del programma.
Le linee 1710-1729 usano espressioni booleane per verificare se il blip ha colpito una delle pareti del riquadro. La linea 1710 controlla le pareti laterali, la 1720 controlla le pareti in alto o in basse. Se non è stata colpita alcuna parete, il programma torna all'inizio del ciclo di movimento alla linea 1570.

In caso di urto

Il modulo successivo, le linee 1790-1900, entra in azione quando il blip urta contro qualcosa, modificandone il movimento, producendo un effetto sonoro e modificando il colore del blip.
Se il blip ha colpito una parete laterale, la linea 1790 inverte il suo movimento orizzontale; se invece ha colpito la parete superiore o quella inferiore, la linea 1800 inverte il suo movimento verticale.
Le linee 1820-1840 ci danno un effetto sonoro. La 1820 rilascia qualunque suono precedente; la 1830 sceglie a caso una regolazione in frequenza e la inserisce nell'opportuno registro di SID; la 1840 poi innesca il suono.
Infine, le linee 1860-1880 modificano il colore del blip, il quale passerà progressivamente attraverso tutta la serie dei colori degli sprite, nero escluso. Dopo aver risolto tutti i problemi relativi a un urto contro una parete, il programma salta alla linea 1570, che è la prima del ciclo di movimento.

Per pulire

Il modulo finale di "Rimbalzo" spegne il suono e lo sprite e quindi pulisce lo schermo direttamente. Per essere un po' più precisi, si può pulire tutti i registri di SID e di VIC usati nel programma.

ORGANINO

Il prossimo programma usa la grafica a caratteri complessi e una esplorazione più veloce della tastiera per creare uno strumento musicale animato. Il programma è riportato nella figura 10.3: battetelo, salvatelo e fatelo girare. Quando suonate lo strumento, le note avranno la durata della pressione del tasto.

Stringhe di bit

Questo programma usa lunghe stringhe di caratteri per disegnare velocemente i tasti. Le stringhe contengono caratteri di controllo del cursore, caratteri di opzioni di visualizzazione, caratteri grafici e caratteri di testo. Queste stringhe richiedono tempo per essere fissate, ma rendono semplice la programmazione e veloce la visualizzazione.
I primi moduli di "Pianorgano" costruiscono sedici stringhe di caratteri per visualizzare i tasti dello strumento. Vi sono due stringhe per ciascuno degli otto tasti, una con una bocca chiusa e l'altra con una bocca aperta.
Le linee 1050-1070 inizializzano due stringhe di tabulazioni. D$ contiene un comando HOME e 23 comandi "cursore giù". R$ contiene 40 comandi "cursore a destra". Usando queste stringhe in combinazione con la funzione LEFT$, possiamo far spostare il cursore ovunque sullo schermo.
Le linee 1120-1210 costruiscono otto stringhe con la bocca chiusa. Dapprima le linee 1120-1140 costruiscono una sezione comune a tutte le otto stringhe. La linea 1150 prepara un pezzo che finirà tutte le otto stringhe, poi le linee 1160-1210 assemblano le otto stringhe personalizzate.
La linea 1170 aggiunge i pezzi di D$ e R$ che porteranno il cursore alla giusta posizione di partenza sullo schermo. Le otto immagini avranno in comune la stessa posizione verticale. Ciascuna, tuttavia, avrà una diversa posizione orizzontale.
La linea 1180 aggiunge la sezione comune costruita nelle linee 1120-1140. Poi la 1190 usa un piccolo trucco per aggiungere un numero a ciascuna immagine. I tasti che cantano hanno codici numerici: da 1 a 8. Quando sono premuti i tasti sulla tastiera da 1 a 8, entrerà in azione il tasto corrispondente dell'organino. I codici di caratteri per i numeri vanno da 48 a 57. La linea 1190 non fa altro che sommare il valore della variabile di controllo N a 48, poi usa la funzione CHR$ per produrre il carattere che corrisponde al valore di N. Per esempio, se N ha il valore

```
1000 REM *** PIANORGANO ***
1010 :
1020 :
1030 REM ** FISSA LE STRINGHE DI TAB
1040 :
1050 D$ = "[illegible]"
1060 R$ = "[illegible]"
1070 R$ = R$ + R$
1080 :
1090 :
1100 REM ** CREA LE STRINGHE DELLE BOCCHE CHIUSE
1110 :
1120 CM$ = "    [illegible]- -[illegible] - [illegible]"
1130 CM$ = CM$ + " - [illegible]"
1140 CM$ = CM$ + "    [illegible] "
1150 FP$ = " [illegible]    "
1160 FOR N = 1 TO 8
1170 :   CM$(N) = LEFT$(D$,4) + LEFT$(R$, 5*N - 4)
1180 :   CM$(N) = CM$(N) + CM$
1190 :   CM$(N) = CM$(N) + CHR$(48 + N)
1200 :   CM$(N) = CM$(N) + FP$
1210 NEXT N
1220 :
1230 :
1240 REM ** CREA LE STRINGHE DELLE BOCCHE APERTE
1250 :
1260 PM$ = "[illegible] O[illegible] - [illegible] ● [illegible]"
1270 PM$ = PM$ + "[illegible] "
1280 PM$ = PM$ + "[illegible]    [illegible] "
1290 FOR N = 1 TO 8
1300 :   PM$(N) = LEFT$(D$,4) + LEFT$(R$, 5*N - 4)
1310 :   PM$(N) = PM$(N) + PM$
1320 :   PM$(N) = PM$(N) + CHR$(48 + N)
1330 :   PM$(N) = PM$(N) + FP$
1340 NEXT N
1350 :
1360 :
1370 REM ** FISSA I CODICI DI COLORE
1380 :
1390 FOR N = 1 TO 8   :REM COLORA I TASTI
1400 :    READ HU(N)
1410 NEXT N
1420 :
1430 DATA 14, 4, 3, 7, 12, 5, 8, 1
1440 :
1450 :
1460 REM ** INIZIALIZZA IL SID E LE FREQUENZE
1470 :
1480 SID = 54272      :REM CHIP SONORO
1490 POKE SID+3, 4    :REM AMP. IMPULSO
1500 POKE SID+5, 10   :REM ATT=0, DEC=10
1510 POKE SID+6, 169  :REM SOS=10, RIL=9
1520 POKE SID+24,15   :REM VOLUME MAX
1530 WF = 64          :REM FORMA D'ONDA IMPULSO
1540 :
1550 FOR N=1 TO 8     :REM FISSA I VALORI
1560 :    READ FH(N)  :REM DI FREQUENZA
1570 :    READ FL(N)  :REM PER 8 NOTE
1580 NEXT N
1590 :
1600 DATA 8, 98, 9, 104
1610 DATA 10, 143, 11,  48
1620 DATA 12, 143, 14,  25
1630 DATA 15, 210, 16, 195
1640 :
1650 :
```

```
1660 REM ** FISSA I COLORI DELLO SCHERMO
1661 REM  - TUTTI I TASTI RIPETONO
1662 REM -  VELOCIZZA SCANSIONE TASTIERA
1670 :
1680 POKE 53280, 0   :REM BORDO NERO
1690 POKE 53281, 0   :REM SFONDO NERO
1700 POKE 650, 128   :REM RIPETIZ TASTI
1710 POKE 56325, 20  :REM SCANS VELOCE
1720 :
1730 :
1740 REM ** 8TAMPA 8 BOCCHE CHIUSE
1750 :
1760 PRINT "□";         :REM PULISCE SCHERMO
1770 PRINT "▓"          :REM GRIGIO SCURO
1780 FOR N = 1 TO 8
1790 :    PRINT CM$(N)    :REM LE BOCCHE
1800 NEXT N
1810 PRINT "█"          :REM BIANCO
1620 :
1830 :
1840 REM ** STAMPA IL PRONTO
1850 :
1860 PRINT LEFT$(D$,18); SPC(9);
1870 PRINT "PREMI I TASTI █1▀-█8▀ PER SUONARE"
1880 PRINT : PRINT SPC(9);
1890 PRINT "PREMI █LO SPAZIO▀ PER FINIRE"
1900 :
1910 :
1920 REM ** ESPLORA LA TASTIERA
1930 :
1940 GET KP$
1950 IF KP$ = "" THEN 1940
1960 IF KP$ = " " THEN 2200
1970 KP = VAL (KP$)
1980 IF KP<1 OR KP>8 THEN 1940
1990 :
2000 :
2010 REM ** SUONA UNA NOTA
2020 :
2030 POKE 646, HU(KP)   :REM FISSA CAR
2040 PRINT PM$(KP)      :REM BOCCA APERTA
2050 POKE SID+1,FH(KP)  :REM FISSA FREQ
2060 POKE SID,FL(KP)    :REM FISSA FREQ
2070 POKE SID+4, WF+1   :REM INIZIO SUONO
2080 :
2090 GET KP$ :REM SUONA FINCHE' IL TASTO NON E' RILASCIATO
2100 IF VAL(KP$) = KP THEN 2090
2110 :
2120 POKE 646, 11      :REM NERO A GRIGIO
2130 PRINT CM$(KP)     :REM BOCCA CHIUSA
2140 POKE SID+4, WF    :REM FINE SUONO
2150 GOTO 1950         :REM ESPLORA ANCORA
2160 :
2170 :
2180 REM ** PULISCE E TORNA A POSTO
2190 :
2200 POKE 56325, 66    :REM REGOLA LA SCANSIONE TASTIERA
2210 POKE 646, 1   :REM CAR COLORE BIANCO
2220 PRINT "□";    :REM PULISCE LO SCHERMO
2230 FOR REG=SID TO SID+24    :REM PULISCE
2240 :    POKE REG, 0         :REM IL SID
2250 NEXT REG
2260 :
2270 END
```

FIG. 10.3. Listato del programma "Pianorgano".

4, la linea 1190 sommerà 48, con risultato 52 (che è il codice del numero 4).
Dopo aver preparato le stringhe con la bocca chiusa, le linee 1260-1340 preparano otto stringhe a bocca aperta.
Il procedimento è analogo a quello delle linee 1120-1210: le differenze principali stanno nei particolari dell'immagine. La figura 10.4 presenta le due diverse immagini dei tasti che cantano, l'uno con la bocca chiusa e l'altro con la bocca aperta.
Il modulo finale della sezione immagazzina otto codici di colore nella matrice HU( ). Ricordatevi che i tasti sono numerati da 1 a 8. Il codice del colore di ciascun tasto verrà usato per stabilire il colore dell'immagine a bocca aperta di quel tasto.

Prepariamo SID, lo schermo e la tastiera

Il programma usa la forma d'onda a impulso e un inviluppo ADSR scelto con cura per creare suoni a mezza strada fra quelli di un pianoforte e quelli di un organo. Le linee 1480-1530 regolano i necessari registri del SID.
Le linee 1550-1630 stabiliscono due matrici, FH( ) e FL( ), che conterranno le regolazioni di frequenza per le otto note. I valori negli enunciati DATA sono tratti dall'appendice O: produrranno le note do, re, mi, fa, sol, la e si (C, D, E, F, G, A, B) della terza ottava e il do (C) della quarta ottava.

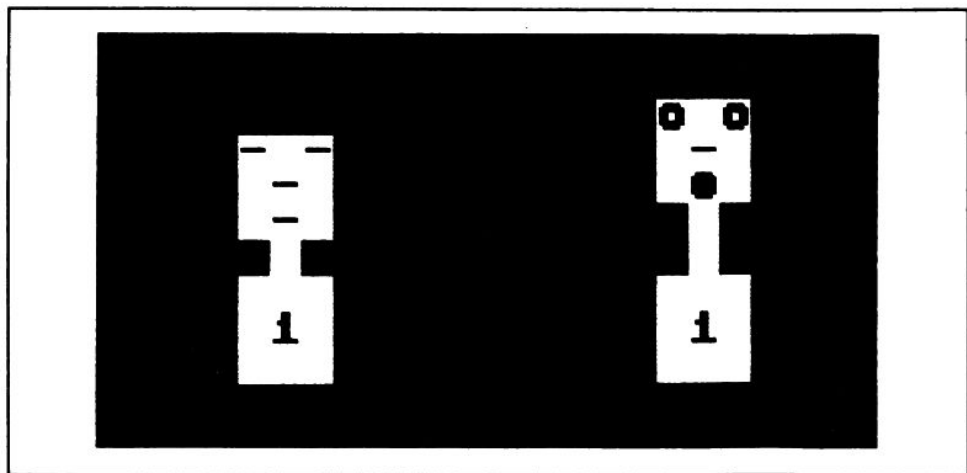



FIG. 10.4. Due immagini di tasti che cantano: con la bocca chiusa e con la bocca aperta.

Le linee 1680 e 1690 scelgono il colore nero per lo sfondo e il bordo dello schermo. Io preferisco nettamente lo sfondo nero, perché mi sembra che i colori risaltino davvero molto bene. In questo programma ho deciso di indulgere al mio gusto.
La linea 1700 sfrutta una particolarità che abbiamo già visto. Quando la locazione di memoria 650 contiene il valore 128, tutti i tasti della tastiera si ripeteranno, se tenuti premuti abbastanza a lungo.
La linea 1710 presenta un nuovo trucco. È molto piacevole lavorare con il Commodore 64, perché si può avere un grande controllo sulla configurazione dell'hardware. Normalmente il Commodore 64 esplora la tastiera alla ricerca di un eventuale tasto premuto 60 volte al secondo. In questo programma, dobbiamo esplorarla più spesso, per avere uno strumento che risponda efficientemente. La locazione di memoria 56325 è un registro che controlla la velocità di esplorazione della tastiera. Normalmente contiene il valore 66, ma inserendovi il valore 20 si può far sì che il calcolatore esplori la tastiera 200 volte al secondo. Alla fine del programma, lo si riporta alla velocità normale. In caso contrario, succederanno cose strane: potete provare, se vi piacciono le stranezze.

La visualizzazione iniziale

I due moduli successivi sono immediati. Le linee 1760-1810 puliscono lo schermo, quindi stampano le otto stringhe a bocca chiusa in grigio scuro. Le 1860-1890 stampano alcune istruzioni per suonare lo strumento.
Ricordate che i caratteri dall'aspetto strano delle linee 1770 e 1810 rappresentano comandi di colore. (Potete consultare l'appendice E, se non vi ricordate più quali sono.)

Il ciclo principale del programma

Siamo arrivati al ciclo principale. Le linee 1940-1980 esplorano la tastiera. Premendo la barra spazio il programma si conclude, mentre i numeri da 1 a 8 innescheranno una nota; ogni altro tasto verrà ignorato.
Le linee 2030-2150 suonano una nota. Questa sezione del programma è relativamente breve e semplice, grazie a tutto il lavoro di preparazione effettuato prima dal programma. La linea 2030 avvia il processo fissando un nuovo colore. Il sistema operativo del Commodore usa la locazione 646 per stabilire il colore in cui deve disegnare i caratteri. Poi la linea 2040 disegna un'immagine con la bocca aperta. Il colore e la stringa a

bocca aperta corrispondono al numero del tasto che è stato premuto. Le linee 2050-2060 poi fissano la frequenza della nota, e la linea 2070 innesca il suono.
L'inviluppo ADSR per i suoni del nostro organino ha un attacco veloce, un decadimento abbastanza lento e un livello di sostegno pari a circa due terzi del volume massimo. La velocità di rilascio è molto simile a quella di attacco. Se una nota viene tenuta per poco, suonerà come una nota di pianoforte; quanto più a lungo viene tenuta, tanto più assomiglierà a una nota d'organo.
Le linee 2090-2100 costituiscono il motivo per cui abbiamo reso più veloce l'esplorazione della tastiera.
La 2090 riceve l'informazione di un tasto premuto e la immagazzina nella variabile KP$. Se un tasto viene tenuto premuto, il valore di KP$ corrisponderà a quello di KP, il numero della nota che in quel momento viene suonata. In tal caso, il programma esegue una rapida «conversione a U» e torna alla 2090 per leggere nuovamente la tastiera. Non appena il tasto viene lasciato, il controllo di corrispondenza della linea 2100 fallirà e il programma passerà a terminare la nota. Con una velocità di esplorazione della tastiera normale, queste due linee non funzionerebbero correttamente: la procedura di ricezione del carattere richiede troppo tempo, e si perderebbero molti movimenti dei tasti. La velocità più elevata risolve il problema.
Le quattro linee successive concludono la nota. La 2120 riporta il colore al grigio scuro. La 2130 disegna l'opportuna immagine a bocca chiusa. La 2140 rilascia il suono e la 2150 salta indietro alla 1950 per controllare se è stato premuto un nuovo tasto.

Qualche riflessione conclusiva

Come abbiamo già detto, premendo la barra spazio il programma termina. Le linee 2200-2250 puliscono tutto.
La 2200 riporta al valore normale la velocità di esplorazione della tastiera; la 2210 riporta il colore dei caratteri al bianco; la 2220 pulisce lo schermo e le 2230-2250, per amor di perfezione, riportano ai valori normali i primi 24 registri del SID.
Vi sono parecchie cose che potete tentare di fare, con questo programma. Potete aggiungere altri tasti allo strumento, potete usare immagini diverse, aggiungere più voci, modificare lo stile dell'animazione o variare il meccanismo della tastiera. La Commodore ha dotato la sua macchina di hardware eccellente: con un software ben progettato, potete creare strumenti musicali animati mai visti o sentiti prima.

IL COORDINAMENTO FRA SUONO E IMMAGINE

Esiste un bellissimo film di Charlie Chaplin che dovete assolutamente vedere, se avete qualche interesse per il coordinamento fra suoni e immagini: è *Luci della città*. Chaplin era diventato espertissimo nell'epoca del film muto: era diventato così bravo che sembrava quasi di sentire i suoni, nei suoi film. *Luci della città* fu uno dei suoi primi film sonori.

In questa pellicola il suono è usato con parsimonia, molto oculatamente, e con grande effetto. Chaplin era un maestro nel coordinamento comico e drammatico, e riuscì a trasferire quella sua abilità nel lavoro con i suoni. Spesso un suono arriva prima di quel che ci si aspetti, preannunciando un'azione a venire. A volte arriva un po' in ritardo, aumentando il pregio di una scena. Usa il suono con parsimonia, per non ottundere il gusto della platea.

Il coordinamento di suoni e immagini non deve essere necessariamente perfetto. Spesso, anzi, un minimo spostamento può aumentare l'efficacia. Lasciate che la mente degli spettatori lavori un po'. Artisti, maghi e grandi registi cinematografici lo sanno; e anche qualcuno fra i migliori programmatori di computer ha cominciato a imparare gli stessi principi.

Prepariamo stringhe, sprite e suoni

Come gli altri programmi di questo capitolo, anche "Altalena" richiede un po' di preparazione. Ogni elemento viene preparato nel suo modulo.

Le linee 1050-1140 predispongono quattro immagini di ganci: un gancio aperto e uno chiuso per ciascuna delle due posizioni del gancio. Ogni immagine di gancio è una lunga stringa, costruita con un po' tutti i caratteri più strani dell'arsenale Commodore: caratteri di variazione di colore, controlli di cursore, opzioni di visualizzazione, caratteri grafici.

Le parti comuni a tutti e quattro i ganci sono costruite e poi assemblate dalle linee 1110-1140 nelle quattro stringhe.

Tecniche analoghe sono usate nelle linee 1190-1230 per preparare le due immagini di altalena. C'è voluta un po' di sperimentazione per trovare i tasti che creassero pezzi di linea che salgono e scendono gradualmente. Come per le immagini dei ganci, nelle stringhe sono inclusi comandi di controllo del cursore e comandi di controllo del colore; collocare l'altalena nella giusta posizione sullo schermo diventa un gioco da bambini.

Gli stessi dati sono usati per creare ambedue gli sprite. Le linee 1280-1310 caricano i dati, i quali sono memorizzati nelle linee 1330-1430.

```
1000 REM *** ALTALENA ***
1010 :
1020 :
1030 REM ** FISSA LE STRINGHE DEI GANCI
1040 :
1050 H1$ = "
1060 H1$ = H1$ + "
1070 H2$ = "
1080 H2$ = H2$ + "
1090 PL$ = "
1100 R$ = "
1110 PH$(1,1) = PL$ + H1$
1120 PH$(1,2) = PL$ + H2$
1130 PH$(2,1) = PL$ + R$ + H1$
1140 PH$(2,2) = PL$ + R$ + H2$
1150 :
1160 :
1170 REM ** FISSA LE STRINGHE DELL'ALTALENA
1180 :
1190 SS$(1) = "
1200 SS$(2) = "
1210 T$ = "
1220 SS$(1) = T$ + SS$(1)
1230 SS$(2) = T$ + SS$(2)
1240 :
1250 :
1260 REM ** CARICA L'IMMAGINE DELLO SPRITE
1270 :
1280 FOR N = 12288 TO 12350
1290 :    READ SPDTA
1300 :    POKE N, SPDTA
1310 NEXT N
1320 :
1330 DATA   0, 255,   0,   1, 129, 128
1340 DATA   3,   0, 192,   6,   0,  96
1350 DATA  12,   0,  48,  24, 231,  24
1360 DATA  48, 165,  12,  32, 231,   4
1370 DATA  32,   0,   4,  32,  36,   4
1380 DATA  38,  60, 100,  35, 129, 196
1390 DATA  48, 231,  12,  24,  60,  24
1400 DATA  14,   0, 112,   3, 255, 192
1410 DATA   0, 129,   0,   0, 129,   0
1420 DATA   0, 129,   0,   0, 129,   0
1430 DATA   3, 231, 192
1440 :
1450 :
1460 REM ** STAMPA IL PRONTO
1470 :
1480 POKE 53281, 0      :REM SFONDO NERO
1490 PRINT "
1500 PRINT SPC(10); "PREMI A ";
1510 PRINT "PER L'AZIONE"
1520 PRINT
1530 PRINT SPC(9); "PREMI ";
1540 PRINT "BARRA SPAZIO PER FINIRE"
1550 :
1560 :
1570 REM ** REGOLA GLI SPRITE
1580 :
1590 VIC = 53248        :REM CHIP GRAFICO
1600 POKE 2040, 192 :REM PUNTATORE SPRITE 0
1610 POKE 2041, 192 :REM PUNTATORE SPRITE 1
1620 :
1630 POKE VIC, 92    :REM SPR0 INIZ POS ORIZ
```

```
1640 POKE VIC+1, 77 :REM SPR0 INIZ POS VERT
1650 POKE VIC+2, 220 :REM SPR1 INIZ POS ORIZ
1660 POKE VIC+3, 150 :REM SPR1 INIZ POS VERT
1670 :
1680 POKE VIC+39, 4 :REM SPR0 INIZIA PORPORA
1690 POKE VIC+40, 3 :REM SPR1 INIZIA CYAN
1700 POKE VIC+23, 3 :REM ESPANSIONE VERT
1710 POKE VIC+29, 3 :REM ESPANSIONE ORIZ
1720 :
1730 POKE VIC+21, 3 :REM SPRITE 0-1 ATTIVATI
1740 :
1750 :
1760 REM ** INIZIALIZZA IL SID
1770 :
1780 SID = 54272         :REM CHIP SONORO
1790 FOR REG = SID TO SID+24
1800 :   POKE REG, 0     :REM PULISCE
1810 NEXT REG
1820 POKE SID+24, 15     :REM VOLUME MAX
1830 :
1840 :
1850 REM ** PREPARA LA VOCE 1 PER GONG
1860 :
1870 POKE SID+1, 5    :REM V-1 FREQ
1880 POKE SID+5, 11   :REM ATT=0, DEC=11
1890 POKE SID+6, 10   :REM SOS=0, RIL=10
1900 :
1910 :
1920 REM ** REGOLA LA VOCE 2 PER IL FISCHIO
1930 :
1940 POKE SID+12, 12 :REM ATK=0, DKY=12
1950 :
1960 :
1970 REM ** REGOLA LA VOCE 3 PER IL CLIC DEL GANCIO
1980 :
1990 POKE SID+15, 21 :REM V-3 FREQ
2000 POKE SID+20,192 :REM SST=12, RLS=0
2010 :
2020 :
2030 REM ** INIZIALIZZA GANCI E ALTALENA
2040 :
2050 FH = 1             :REM GANCIO 1 PIENO
2060 EH = 2             :REM GANCIO 2 VUOTO
2070 PRINT PH$(FH, 1) :REM STAMPA GANCIO 1
2080 PRINT PH$(EH, 2) :REM STAMPA GANCIO 2
2090 PRINT SSW$(1)     :REM STAMPA ALTALENA
2100 :
2110 :
2120 REM ** ESPLORAZIONE TASTIERA
2130 :
2140 GET KP$
2150 IF KP$ = "" THEN 2140  :REM ESPLORAZIONE
2160 IF KP$ = "A" THEN 2230 :REM AZIONE
2170 IF KP$ = " " THEN 2980 :REM FINISCE
2180 GOTO 2140 :REM GLI ALTRI TASTI VENGONO FILTRATI
2190 :
2200 :
2210 REM ** LIBERA UNO SPRITE
2220 :
2230 POKE SID+18,129  :REM INIZIO CLIC
2240 PRINT PH$(FH,2)  :REM IL GANCIO SI APRE
2250 FOR DL = 1 TO 40 : NEXT DL
2260 POKE SID+18, 128  :REM FINE CLIC
2270 POKE VIC + FH + 38, 3 :REM DIVENTA CYAN
2280 :
2290 :
```

```
2300 REM ** LO SPRITE LIBERATO CADE
2310 :
2320 POKE SID+8,80    :REM V-2 INIZ FRQ
2330 POKE SID+11,17   :REM FISCHIO ATTIVATO
2340 FOR N = 78 TO 145
2350 :    POKE VIC+(FH*2)-1, N :REM CADE
2360 :    POKE SID+8, 158 - N  :REM FISCHIO
2370 NEXT N
2380 POKE SID+11,16   :REM IL FISCHIO SI SPEGNE
2390 :
2400 :
2410 REM ** MOVIMENTO ALTALENA
2420 :
2430 POKE SID+4,21      :REM INIZIO GONG
2440 PRINT SSW$(3-FH)  :REM MOVIMENTO ALTALENA
2450 POKE VIC+(FH*2)-1,150 :REM SPOSTAMENTO
2460 POKE VIC+(EH*2)-1,146 :REM DEGLI SPRITE
2470 POKE SID+4,20      :REM IL GONG SI ESTINGUE
2480 :
2490 :
2500 REM ** FA VIBRARE LO SPRITE CADUTO
2510 :
2520 HR = VIC + (FH*2) - 2 :REM REG OR
2530 HP = PEEK (HR)        :REM POS OR
2540 CR = VIC + FH + 38    :REM REG COLORE
2550 FOR VB = 1 TO 5       :REM 6 VIBRAZIONI
2560 :    POKE HR, HP - 4  :REM A SINISTRA
2570 :    POKE CR, 1       :REM DIVENTA BIANCO
2580 :    POKE SID+1, 6    :REM ALTA FREQ
2590 :    POKE HR, HP      :REM AL CENTRO
2600 :    POKE CR, 2       :REM DIVENTA ROSSO
2610 :    POKE SID+1, 4    :REM BASSA FREQ
2620 :    POKE HR, HP + 4  :REM A DESTRA
2630 :    POKE CR, 7       :REM DIVENTA GIALLO
2640 :    POKE SID+1, 5    :REM MEDIA FREQ
2650 NEXT VB
2660 POKE HR, HP   :REM RIPRISTINA POSIZIONE
2670 POKE CR, 3    :REM RIPRISTINA COLORE
2680 :
2690 :
2700 REM ** LO SPRITE SULL'ALTALENA SALE
2710 :
2720 POKE SID+8,80     :REM V-2 INIZ FRQ
2730 POKE SID+11,17    :REM FISCHIO ATTIVATO
2740 FOR N = 145 TO 77 STEP -1
2750 :    POKE VIC+(EH*2)-1, N :REM SALE
2760 :    POKE SID+8, 158 - N  :REM FISCHIO
2770 NEXT N
2780 POKE SID+11,16    :REM IL FISCHIO SI ESTINGUE
2790 :
2800 :
2810 REM ** CATTURA DI UNO SPRITE
2820 :
2830 POKE SID+18,129    :REM INIZIO CLIC
2840 PRINT PH$(EH,1)    :REM IL GANCIO SI CHIUDE
2850 FOR DL = 1 TO 40 : NEXT DL
2860 POKE SID+18, 128  :REM FINE CLIC
2870 POKE VIC + EH + 38, 4 :REM DIVENTA ROSSO
2880 :
2890 :
2900 REM ** COMMUTA TRA GANCIO PIENO (FH)
2901 REM     E VUOTO (EH) E RITORNA
2910 :
2920 TEMP = FH : FH = EH : EH = TEMP
2930 GOTO 2140
```

```
2940 ⏎
2950 :
2960 REM ** FINISCE, PULISCE, TORNA AL POSTO
2970 :
2980 POKE VIC+21,0 :REM SPRITE DISATTIVATI
2990 POKE SID+24,0 :REM VOLUME SPENTO
3000 POKE VIC+23,0 :REM ESPANSIONE VERT DISATTIVATA
3010 POKE VIC+29,0 :REM ESPANSIONE ORIZ DISATTIVATA
3020 PRINT "♥";    :REM PULISCE LO SCHERMO
3030 :
3040 END
```

FIG. 10.5. Listato del programma "Altalena".

Le linee 1480-1540 stampano i caratteri di pronto sullo schermo: nulla di problematico. Poi le linee 1590-1730 danno le regolazioni iniziali del VIC per gli sprite. Anziché calcolare l'esatta posizione degli sprite, ho cominciato con una stima e poi ho usato tecniche di ricerca intelligente (prova ed errore) per arrivare ai valori giusti.
Le immagini sono pronte, ed è il momento di preparare i suoni. La prima voce del SID verrà usata per il gang; la seconda darà il sibilo del volo, e la terza creerà il rumore dei ganci. Le linee 1780-1820 puliscono i 24 registri importanti del SID e fissano il volume al massimo. Poi le linee 1870-2000 inseriscono i valori necessari per produrre i tre suoni.
Una volta che il programma è avviato, vengono usate due variabili per tener traccia della situazione dei ganci e delle creature. FH conterrà il numero del gancio che tiene una creatura, e EH conterrà il numero del gancio vuoto. Il gancio 1 e la creatura 1 sono a sinistra; gancio 2 e creatura 2 sono a destra.
Le linee 2050-2060 inizializzano queste variabili, poi le linee 2070-2090 disegnano le opportune immagini di gancio e altalena. A questo punto tutto il palcoscenico è pronto.

Articolazione dell'azione

Le linee 2140-2180 formano un modulo, ormai familiare, di esplorazione della tastiera. Vengono ignorati tutti i tasti, fatta eccezione per A e la barra spazio.
Premendo A si avvia un ciclo d'azione; premendo la barra spazio il programma termina.
Il ciclo d'azione si suddivide in sei moduli. Nel primo, la creatura appesa al gancio viene liberata. Nel secondo cade emettendo un fischio. Nel terzo colpisce l'altalena, che muta la sua posizione, insieme con le due creature. Nel quarto, la creatura appena caduta vibra. Nel quinto l'altro

sprite sale in aria, fischiando. Nel sesto lo sprite in salita viene bloccato dal suo gancio.
Le linee 2230-2270 si preoccupano di liberare uno sprite. Comincia il rumore del gancio, il gancio si apre, c'è un piccolo ritardo, poi il rumore finisce e lo sprite cambia colore.
Le linee 2320-2380 fanno cadere lo sprite. Viene fissata la frequenza del suono iniziale, e parte il fischio. Lo sprite cade grazie a un ciclo che lo fa scendere, facendo diminuire la frequenza del suono in parallelo. Alla base, il fischio cessa (grazie a una attenta scelta del decadimento, durante il viaggio anche il volume del suono è andato progressivamente diminuendo).
Poi lo sprite in caduta raggiunge l'altalena, e siamo pronti per la terza parte della sequenza. Inizia un suono di gong; l'altalena gira; lo sprite si muove; e il rumore di gong comincia lentamente a svanire. Tutto questo avviene nelle linee 2430-2470.
Le linee 2520-2670 fanno vibrare lo sprite caduto. Parallelamente allo spostarsi su e giù della frequenza del gong, lo sprite si sposta avanti e indietro, in orizzontale, e muta di colore. Questa attività viene ripetuta varie volte. Infine, quando il suono del gong svanisce, la creatura si ferma e il suo colore ritorna azzurro.
Eccoci al quinto modulo del ciclo di azione. L'altro sprite sale in aria. Confrontate le linee 2720-2780 con le 2320-2380, che facevano cadere lo sprite dal gancio. I due moduli sono molto simili. Innanzitutto, si dà una frequenza iniziale alla voce 2. Poi si avvia il suono. Viene poi il ciclo principale del modulo. Man mano che lo sprite si sposta in su lungo lo schermo, sale la frequenza della voce 2. Infine, quando arriva in cima, il fischio viene rilasciato.
Nella sesta parte, lo sprite in ascesa viene catturato, così come nella prima parte uno sprite veniva lasciato libero. Il tutto si verifica alle linee 2830-2870.
Comincia il rumore del gancio; il gancio si chiude a scatto; c'è un po' di ritardo; il rumore del gancio termina; lo sprite viene privato di ogni colore di libertà.
L'azione è finita, e le condizioni dei due sprite sono state scambiate. Il gancio che era vuoto ora è occupato, e viceversa. La linea 2920 aggiorna le variabili EH e FH sulla base di questi fatti, e la linea 2930 ci riporta a leggere nuovamente la tastiera.

Pulizia

Le linee 2890-3020 effettuano una normale operazione di pulizia. Potete scegliere di essere più precisi per quanto riguarda i vari registri del SID e del VIC.

Quando ho scritto questo programma, ho realizzato prima le linee generali dell'azione, e ho lasciato per ultimo il perfezionamento nei dettagli dei suoni e dei movimenti degli sprite. Con "Altalena" questo metodo di risoluzione dei problemi ha funzionato molto bene.

QUALCHE RIFLESSIONE FINALE

Prima di passare agli esercizi, ecco alcune cose da tenere a mente quando si uniscono suoni e grafica:

- temporizzazione: un effetto molto semplice può avere un grande impatto, se arriva al momento giusto;
- perfezionamento: quando ogni elemento si inquadra perfettamente nell'effetto complessivo, la sinergia è massima;
- semplicità: eliminate ogni orpello; ciascun suono, ciascuna immagine deve servire a uno scopo preciso;
- unità di progetto: i singoli elementi debbono essere di sostegno reciproco.

CHE COSA ABBIAMO IMPARATO

In questo capitolo abbiamo esplorato tre programmi che uniscono suoni e grafica. Più specificamente, abbiamo visto:

- Come creare sinergia, perché l'effetto di grafica e suoni combinati sia maggiore della somma delle singole parti
- Quali tecniche siano utili per risolvere compiti di programmazione complessi
- Il programma "Rimbalzo", in cui sono combinati la grafica a caratteri e a sprite e effetti sonori semplici, e in cui viene introdotta una semplice tecnica di rimbalzo
- Il programma "Pianorgano", in cui vengono utilizzate stringhe di caratteri complesse e una esplorazione più veloce della tastiera, per creare uno strumento musicale animato
- Come si coordinano suoni e immagini in modo raffinati e artistici
- Il programma "Altalena", con un insieme complesso di azioni che coinvolgono tutte e tre le voci del SID, due sprite e stringhe di caratteri complesse.

Spero che la nostra esplorazione delle capacità grafiche e sonore del Commodore 64 vi sia stata utile.
Siate sempre curiosi, continuate a studiare, e divertitevi!

ESERCIZI

Test

1. Quando il tutto è maggiore della somma delle parti, si può parlare di ....
2. La tecnica della suddivisione di un compito complesso di programmazione in parti sempre più semplici prende il nome di ....
3. Le parti di una immagine che stanno ferme sono ....; le parti che si muovono sono ....
4. Il programma "Rimbalzo" usa espressioni .... per controllare le collisioni fra blip e parete.
5. Rendendo più veloce la scansione .... nel programma "Pianorgano" si ottiene uno strumento musicale dalla risposta più pronta.
6. In "Altalena", il ciclo complesso dell'azione è stato suddiviso in .... moduli più semplici.

Esercizi di programmazione

1. Modificate il programma "Rimbalzo" in modo che produca rumori con un andamento più regolare, quando lo sprite urta contro una parete.
2. Modificate il programma "Pianorgano" in modo che le teste brillino di colore quando cantano.
3. Modificate il programma "Altalena", così che gli sprite possano muoversi verticalmente, oltre che orizzontalmente, quando colpiscono l'altalena.

Risposte al test

1. sinergia
2. programmazione strutturata "top-down"
3. gli elementi statici; gli elementi dinamici
4. booleane
5. della tastiera
6. sei.

Possibili soluzioni agli esercizi di programmazione

1. Caricate il programma “Rimbalzo”, poi battete queste linee:

```
1000 REM *** ALTRO RIMBALZO ***
1463 :
1465 FQ = 10  :REM FREQUENZA DI INIZIO
1468 FC = 1.3 :REM FATTORE DI CAMBIAMENTO FREQUENZA
1812 FQ = FQ * FC
1814 IF FQ > 100 THEN FC = 0.6
1816 IF FQ < 10  THEN FC = 1.3
1818 :
1830 POKE SID+1, FQ
```

2. Caricate il programma “Pianorgano”, poi battete queste linee:

```
1000 *** ARCOBORGANO ***
1275 JM$ = PM$
1365 :  JM$(N) = PM$(N) + JM$
2093 POKE 646,((PEEK(646)+1)AND 15)OR 1
2096 PRINT JM$(KP)
```

3. Caricate il programma “Altalena”, poi battete queste linee:

```
1000 *** ANCORA ALTALENA ***
2533 VR = HR + 1
2535 VP = PEEK (VR)
2645 :   POKE VR, VP - VB*2
2740 FOR N = 145 TO 77 STEP -1.6
2765 :   POKE VR, VP+(N>115)*(N/3-38)
```

Appendici

APPENDICE A

Disposizione dei registri del VIC

L'indirizzo di partenza del VIC è 53248 ($D000)

Numero del registro - Decimale	Esadecimale	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Questo registro controlla:
0	$00	S0 H7	S0 H6	S0 H5	S0 H4	S0 H3	S0 H2	S0 H1	S0 H0	Posizione orizzontale dello sprite 0
1	$01	S0 V7	S0 V6	S0 V5	S0 V4	S0 V3	S0 V2	S0 V1	S0 V0	Posizione verticale dello sprite 0
2	$02	S1 H7	S1 H6	S1 H5	S1 H4	S1 H3	S1 H2	S1 H1	S1 H0	Posizione orizzontale dello sprite 1
3	$03	S1 V7	S1 V6	S1 V5	S1 V4	S1 V3	S1 V2	S1 V1	S1 V0	Posizione verticale dello sprite 1
4	$04	S2 H7	S2 H6	S2 H5	S2 H4	S2 H3	S2 H2	S2 H1	S2 H0	Posizione orizzontale dello sprite 2
5	$05	S2 V7	S2 V6	S2 V5	S2 V4	S2 V3	S2 V2	S2 V1	S2 V0	Posizione verticale dello sprite 2
6	$06	S3 H7	S3 H6	S3 H5	S3 H4	S3 H3	S3 H2	S3 H1	S3 H0	Posizione orizzontale dello sprite 3
7	$07	S3 V7	S3 V6	S3 V5	S3 V4	S3 V3	S3 V2	S3 V1	S3 V0	Posizione verticale dello sprite 3
8	$08	S4 H7	S4 H6	S4 H5	S4 H4	S4 H3	S4 H2	S4 H1	S4 H0	Posizione orizzontale dello sprite 4
9	$09	S4 V7	S4 V6	S4 V5	S4 V4	S4 V3	S4 V2	S4 V1	S4 V0	Posizione verticale dello sprite 4
10	$0A	S5 H7	S5 H6	S5 H5	S5 H4	S5 H3	S5 H2	S5 H1	S5 H0	Posizione orizzontale dello sprite 5
11	$0B	S5 V7	S5 V6	S5 V5	S5 V4	S5 V3	S5 V2	S5 V1	S5 V0	Posizione verticale dello sprite 5
12	$0C	S6 H7	S6 H6	S6 H5	S6 H4	S6 H3	S6 H2	S6 H1	S6 H0	Posizione orizzontale dello sprite 6
13	$0D	S6 V7	S6 V6	S6 V5	S6 V4	S6 V3	S6 V2	S6 V1	S6 V0	Posizione verticale dello sprite 6
14	$0E	S7 H7	S6 H6	S7 H5	S7 H4	S7 H3	S7 H2	S7 H1	S7 H0	Posizione orizzontale dello sprite 7
15	$0F	S7 V7	S7 V6	S7 V5	S7 V4	S7 V3	S7 V2	S7 V1	S7 V1	Posizione verticale dello sprite 7
16	$10	S7 H8	S6 H8	S5 H8	S4 H8	S3 H8	S2 H8	S1 H8	S0 H8	Bit più significativo delle posizioni orizzontali
17	$11	Bit 7 del raster	Modo testo colore esteso	Modo mappa di bit	Schermo vuoto	24 o 25 righe di testo	Bit 2 dello scorrimento verticale	Bit 1 dello scorrimento verticale	Bit 0 dello scorrimento verticale	Funzioni varie

18	$12	Bit 7 del raster	Bit 6 del raster	Bit 5 del raster	Bit 4 del raster	Bit 3 del raster	Bit 2 del raster	Bit 1 del raster	Bit 0 del raster	Registro raster
19	$13	LP H7	LP H6	LP H5	LP H4	LP H3	LP H2	LP H1	LP H0	Posizione orizzontale della penna ottica
20	$14	LP V7	LP V6	LP V5	LP V4	LP V3	LP V2	LP V1	LP V0	Posizione verticale della penna ottica
21	$15	S7 On/off	S6 On/off	S5 On/off	S4 On/off	S3 On/off	S2 On/off	S1 On/off	S0 On/Off	Attiva/disattiva gli sprite
22	$16	—	—	Reset - sempre a 0	Modo multi-colore	38 o 40 colonne di testo	Bit 2 dello scorrimento orizzontale	Bit 1 dello scorrimento orizzontale	Bit 0 dello scorrimento orizzontale	Funzioni varie
23	$17	S7 EV	S6 EV	S5 EV	S4 EV	S3 EV	S2 EV	S1 EV	S0 EV	Espansione dello sprite in verticale
24	$18	Bit 3 dello schermo testo	Bit 2 dello schermo testo	Bit 1 dello schermo testo	Bit 0 dello schermo testo	Bit 2 def. caratteri	Bit 1 def. caratteri	Bit 0 def. caratteri	—	Puntatori di memoria per visualizzazione caratteri, mappa di bit e schermo
25	$19	Interruzioni dal VIC	—	—	—	Penna ottica collegata	Collisione fra sprite	Collisione fra sprite e sfondo	Corrispondenza conteggio raster	Registro delle interruzioni
26	$1A	—	—	—	—	Penna ottica collegata	Collisione fra sprite	Collisione fra sprite e sfondo	Corrispondenza conteggio raster	Attivazione delle interruzioni
27	$1B	S7 SBP	S6 SBP	S5 SBP	S4 SBP	S3 SBP	S2 SBP	S1 SBP	S0 SBP	Priorità fra sprite e sfondo
28	$1C	S7 MCM	S6 MCM	S5 MCM	S4 MCM	S3 MCM	S2 MCM	S1 MCM	S0 MCM	Selezione del modo multicolore per gli sprite
29	$1D	S7 EH	S6 EH	S5 EH	S4 EH	S3 EH	S2 EH	S1 EH	S0 EH	Espansione dello sprite in orizzontale
30	$1E	S7 SSC	S6 SSC	S5 SSC	S4 SSC	S3 SSC	S2 SSC	S1 SSC	S0 SSC	Collisione fra sprite
31	$1F	S7 SBC	S6 SBC	S5 SBC	S4 SBC	S3 SBC	S2 SBC	S1 SBC	S0 SBC	Collisione fra sprite e sfondo

L'indirizzo di partenza del VIC è 53248 (ÅD000)

Numero del registro Decimale	Esadecimale	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Questo registro controlla:
32	$20	–	–	–	–	Bordo C3	Bordo C2	Bordo C1	Bordo C0	Colore del bordo
33	$21	–	–	–	–	Sfondo 0 C3	Sfondo 0 C2	Sfondo 0 C1	Sfondo 0 C0	Colore 0 dello sfondo
34	$22	–	–	–	–	Sfondo 1 C3	Sfondo 1 C2	Sfondo 1 C1	Sfondo 1 C0	Colore 1 dello sfondo
35	$23	–	–	–	–	Sfondo 2 C3	Sfondo 2 C2	Sfondo 2 C1	Sfondo 2 C0	Colore 2 dello sfondo
36	$24	–	–	–	–	Sfondo 3 C3	Sfondo 3 C2	Sfondo 3 C1	Sfondo 3 C0	Colore 3 dello sfondo
37	$25	–	–	–	–	SMC 0 C3	SMC 0 C2	SMC 0 C1	SMC 0 C0	Sprite multicolore 0
38	$26	–	–	–	–	SMC 1 C3	SMC 1 C2	SMC 1 C1	SMC 1 C0	Sprite multicolore 1
39	$27	–	–	–	–	S0 C3	S0 C2	S0 C1	S0 C0	Colore 0 dello sprite
40	$28	–	–	–	–	S1 C3	S1 C2	S1 C1	S1 C0	Colore 1 dello sprite
41	$29	–	–	–	–	S2 C3	S2 C2	S2 C1	S2 C0	Colore 2 dello sprite
42	$2A	–	–	–	–	S3 C3	S3 C2	S3 C1	S3 C0	Colore 3 dello sprite
43	$2B	–	–	–	–	S4 C3	S4 C2	S4 C1	S4 C0	Colore 4 dello sprite
44	$2C	–	–	–	–	S5 C3	S5 C2	S5 C1	S5 C0	Colore 5 dello sprite
45	$2D	–	–	–	–	S6 C3	S6 C2	S6 C1	S6 C0	Colore 6 dello sprite
46	$2E	–	–	–	–	S7 C3	S7 C2	S7 C1	S7 C0	Colore 7 dello sprite

APPENDICE B

Memoria dello schermo

Indirizzo usuale
1063
1263
1463
1663
1863
2023
Riga
0
5
10
15
20
24
Colonna
0
5
10
15
20
25
30
35
39
Indirizzo usuale
1024
1064
1104
1144
1184
1224
1264
1304
1344
1384
1424
1464
1504
1544
1584
1624
1664
1704
1744
1784
1824
1864
1904
1944
1984

APPENDICE C

Memoria del Colore

APPENDICE D

Codici di visualizzazione sullo schermo

APPENDICE D

Codici di visualizzazione sullo schermo

Codice	Insieme 1	Insieme 2
0	@	@
1	A	a
2	B	b
3	C	c
4	D	d
5	E	e
6	F	f
7	G	g
8	H	h
9	I	i
10	J	j
11	K	k
12	L	l
13	M	m
14	N	n
15	O	o
16	P	p
17	Q	q
18	R	r
19	S	s
20	T	t

Codice	Insieme 1	Insieme 2
128	@	@
129	A	a
130	B	b
131	C	c
132	D	d
133	E	e
134	F	f
135	G	g
136	H	h
137	I	i
138	J	j
139	K	k
140	L	l
141	M	m
142	N	n
143	O	o
144	P	p
145	Q	q
146	R	r
147	S	s
148	T	t

Codice	Insieme 1	Insieme 2
64	─	─
65	♠	A
66	│	B
67	─	C
68	─	D
69	─	E
70	─	F
71	│	G
72	│	H
73	╮	I
74	╰	J
75	╯	K
76	└	L
77	╲	M
78	╱	N
79	┌	O
80	┐	P
81	●	Q
82	─	R
83	♥	S
84	│	T

Codice	Insieme 1	Insieme 2
192	─	─
193	♠	A
194	│	B
195	─	C
196	─	D
197	─	E
198	─	F
199	│	G
200	│	H
201	╮	I
202	╰	J
203	╯	K
204	└	L
205	╲	M
206	╱	N
207	┌	O
208	┐	P
209	●	Q
210	─	R
211	♥	S
212	│	T

Codice	Insieme 1	Insieme 2	Codice	Insieme 1	Insieme 2	Codice	Insieme 1	Insieme 2	Codice	Insieme 1	Insieme 2
21	U	u	149	U	u	85		U	213		U
22	V	v	150	V	v	86		V	214		V
23	W	w	151	W	w	87		W	215		W
24	X	x	152	X	x	88		X	216		X
25	Y	y	153	Y	y	89		Y	217		Y
26	Z	z	154	Z	z	90		Z	218		Z
27	[	[	155	[	[	91			219		
28	£	£	156	£	£	92			220		
29	]	]	157	]	]	93			221		
30	↑	↑	158	↑	↑	94			222		
31	←	←	159	←	←	95			223		
32			160			96			224		
33	!	!	161	!	!	97			225		
34	"	"	162	"	"	98			226		
35	#	#	163	#	#	99			227		
36	$	$	164	$	$	100			228		
37	%	%	165	%	%	101			229		
38	&	&	166	&	&	102			230		
39	'	'	167	'	'	103			231		
40	(	(	168	(	(	104			232		
41	)	)	169	)	)	105			233		
42	*	*	170	*	*	106			234		

Codice	Insieme 1	Insieme 2	Codice	Insieme 1	Insieme 2	Codice	Insieme 1	Insieme 2	Codice	Insieme 1	Insieme 2
43	+	+	171			107			235		
44	,	,	172			108			236		
45	-	-	173			109			237		
46	.	.	174			110			238		
47	/	/	175			111			239		
48	0	0	176			112			240		
49	1	1	177			113			241		
50	2	2	178			114			242		
51	3	3	179			115			243		
52	4	4	180			116			244		
53	5	5	181			117			245		
54	6	6	182			118			246		
55	7	7	183			119			247		
56	8	8	184			120			248		
57	9	9	185			121			249		
58	:	:	186			122			250		
59	;	;	187			123			251		
60	<	<	188			124			252		
61	=	=	189			125			253		
62	>	>	190			126			254		
63	?	?	191			127			255		

APPENDICE E

Icone di visualizzazione

ICONE DI COLORE

Icona	Tasti da premere	Che cosa fa	Icona	Tasti da premere	Che cosa fa
	CTRL-1	Colore del testo nero		C= – 1	Colore del testo arancione
	CTRL-2	Colore del testo bianco		C= – 2	Colore del testo marrone
	CTRL-3	Colore del testo rosso		C= – 3	Colore del testo rosso chiaro
	CTRL-4	Colore del testo cyan		C= – 4	Colore del testo grigio scuro
	CTRL-5	Colore del testo violetto		C= – 5	Colore del testo grigio medio
	CTRL-6	Colore del testo verde		C= – 6	Colore del testo verde chiaro
	CTRL-7	Colore del testo blu		C= – 7	Colore del testo blu chiaro
	CTRL-8	Colore del testo giallo		C= – 8	Colore del testo grigio chiaro

ALTRE ICONE

Icona	Tasti da premere	Che cosa fa	Icona	Tasti da premere	Che cosa fa
	CLR/home	"Home" del cursore		Shift–CLR/home	Pulische lo schermo
	CRSR ↕	Cursore giù		Shift–CRSR ↕	Cursore su
	CRSR ⇄	Cursore a destra		Shift–CRSR ⇄	Cursore a sinistra
	CTRL–9	Reverse attivato		CTRL-0	Reverse disattivato

APPENDICE F

Codici di colore

0 - nero	8 - arancione
1 - bianco	9 - marrone
2 - rosso	10 - rosso chiaro
3 - cyan	11 - grigio scuro
4 - violetto	12 - grigio medio
5 - verde	13 - verde chiaro
6 - blu	14 - blu chiaro
7 - giallo	15 - grigio chiaro

APPENDICE G

Foglio di codificazione per sprite normali

APPENDICE H

Foglio di codificazione per sprite multicolori

APPENDICE I

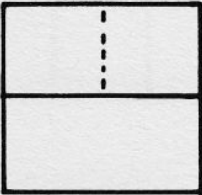
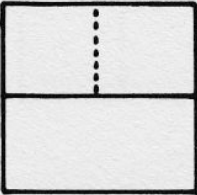
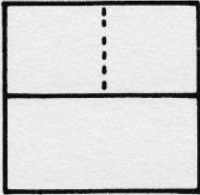
Foglio di codificazione caratteri

Numero del bit	7	6	5	4	3	2	1	0	Numero codici
Valore del bit	128	64	32	16	8	4	2	1	
Byte 0									
Byte 1									
Byte 2									
Byte 3									
Byte 4									
Byte 5									
Byte 6									
Byte 7									

APPENDICE J

Foglio di codificazione dei caratteri multicolori

Valore del bit ▶	128	64	32	16	8	4	2	1	Numero codici ▼
Byte 0									
Byte 1									
Byte 2									
Byte 3									
Byte 4									
Byte 5									
Byte 6									
Byte 7									

Colore 0 dello sfondo (colore dello schermo)

Colore 1 dello sfondo

Colore 2 dello sfondo

Bit inferiori del colore della memoria del colore

APPENDICE K

Foglio di codificazione di blocco carattere 2H × 3V

128 64 32 16 8 4 2 1 128 64 32 16 8 4 2 1

APPENDICE L

Disposizione dei registri del SID

L'indirizzo di partenza del SID è 54272 ($D400)

Numero del registro Decimale	Esadecimale	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Questo registro controlla:	
0	$00	FR7	FR6	FR5	FR4	FR3	FR2	FR1	FR0	Byte basso della frequenza	Voce 1
1	$01	FR15	FR14	FR13	FR12	FR11	FR10	FR9	FR8	Byte alto della frequenza	
2	$02	PW7	PW6	PW5	PW4	PW3	PW2	PW1	PW0	Byte basso della larghezza di impulso	
3	$03	–	–	–	–	PW11	PW10	PW9	PW8	Nibble alto della larghezza di impulso	
4	$04	Rumore	Impulso	Dente di sega	Trian-golare	Prova	Modulaz. ad anello	Sincroniz-zazione	Porta	Controllo di por-ta e forma d'onda	
5	$05	ATK3	ATK2	ATK1	ATK0	DCY3	DCY2	DCY1	DCY0	Attacco/decadimento	
6	$06	SST3	SST2	SST1	SST0	RLS3	RLS2	RLS1	RLS0	Sostegno/rilascio	
7	$07	FR7	FR6	FR5	FR4	FR3	FR2	FR1	FR0	Byte basso della frequenza	Voce 2
8	$08	FR15	FR14	FR13	FR12	FR11	FR10	FR9	FR8	Byte alto della frequenza	
9	$09	PW7	PW6	PW5	PW4	PW3	PW2	PW1	PW0	Byte basso della larghezza di impulso	
10	$0A	–	–	–	–	PW11	PW10	PW9	PW8	Nibble alto della larghezza di impulso	
11	$0B	Rumore	Impulso	Dente di sega	Trian-golare	Prova	Modulaz. ad anello	Sincroniz-zazione	Porta	Controllo di por-ta e forma d'onda	
12	$0C	ATK3	ATK2	ATK1	ATK0	DCY3	DCY2	DCY1	DCY0	Attacco/decadimento	
13	$0D	SST3	SST2	SST1	SST0	RLS3	RLS2	RLS1	RLS0	Sostegno/rilascio	

Numero del registro Decimale	Esadecimale	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Questo registro controlla:	
14	$0E	FR7	FR6	FR5	FR4	FR3	FR2	FR1	FR0	Byte basso della frequenza	Voce 3
15	$0F	FR15	FR14	FR13	FR12	FR11	FR10	FR9	FR8	Byte alto della frequenza	
16	$10	PW7	PW6	PW5	PW4	PW3	PW2	PW1	PW0	Byte basso della larghezza di impulso	
17	$11	–	–	–	–	PW11	PW10	PW9	PW8	Nibble alto della larghezza d'impulso	
18	$12	Rumore	Impulso	Dente di sega	Triangolare	Prova	Modulaz. ad anello	Sincronizzazione	Porta	Controllo di porta e forma d'onda	
19	$13	ATK3	ATK2	ATK1	ATK0	DCY3	DCY2	DCY1	DCY0	Attacco/decadimento	
20	$14	SST3	SST2	SST1	SST0	RLS3	RLS2	RSL1	RLS0	Sostegno/rilascio	
21	$15	–	–	–	–	–	CFR2	CFR1	CFR0	3 bit bassi della frequenza di taglio	Filtro/volume
22	$16	CFR10	CFR9	CFR8	CFR7	CFR6	CFR5	CFR4	CFR3	3 bit alti della frequenza di taglio	
23	$17	RES3	RES2	RES1	RES0	Filtro esterno	Filtro voce 3	Filtro voce 2	Filtro voce 1	Risonanza/filtro	
24	$18	V3 in silenzio	Passa alto	Passa banda	Passa basso	Volume 3	Volume 2	Volume 1	Volume 0	Modo del filtro/volume	
25	$19	GPX 7	GPX 6	GPX 5	GPX 4	GPX 3	GPX 2	GPX 1	GPX 0	Paddle per giochi X	Altro
26	$20	GPY 7	GPY 6	GPY 5	GPY 4	GPY 3	GPY 2	GPY 1	GPY 0	Paddle per giochi Y	
27	$21	V30 7	V30 6	V30 5	V30 4	V30 3	V30 2	V30 1	V30 0	Oscillatore della voce 3	
28	$22	V3E 7	V3E 6	V3E 5	V3E 4	V3E 3	V3E 2	V3E 1	V3E 0	Inviluppo della voce 3	

APPENDICE M

Valori delle note

Ottava	Nome della nota	Frequenza in hertz	Regolaz. di freq. del SID	Byte alto della regol. di freq. del SID	Byte basso della regol. di freq. del SID
0	Do	16.4	269	1	13
0	Do#	17.3	284	1	28
0	Re	18.4	302	1	46
0	Re#	19.4	318	1	62
0	Mi	20.6	338	1	82
0.	Fa	21.8	358	1	102
0	Fa#	23.1	379	1	123
0	Sol	24.5	402	1	146
0	Sol#	26.0	427	1	171
0	La	27.5	451	1	195
0	La#	29.1	477	1	221
0	Si	30.9	507	1	251
1	Do	32.7	536	2	24
1	Do#	34.6	568	2	56
1	Re	36.7	602	2	90
1	Re#	38.9	638	2	126
1	Mi	41.2	676	2	164
1	Fa	43.7	717	2	205
1	Fa#	46.2	758	2	246
1	Sol	49.0	804	3	36
1	Sol#	51.9	851	3	83
1	La	55.0	902	3	134
1	La#	58.3	956	3	188
1	Si	61.7	1012	3	244
2	Do	65.4	1073	4	49
2	Do#	69.3	1137	4	113
2	Re	73.4	1204	4	180
2	Re#	77.8	1276	4	252
2	Mi	82.4	1352	5	72
2	Fa	87.3	1432	5	152
2	Fa#	92.5	1517	5	237
2	Sol	98.0	1608	6	72
2	Sol#	103.8	1703	6	167
2	La	110.0	1804	7	12
2	La#	116.5	1911	7	119
2	Si	123.5	2026	7	234
3	Do	130.8	2146	8	98
3	Do#	138.6	2274	8	226
3	Re	146.8	2408	9	104
3	Re#	155.6	2553	9	249
3	Mi	164.8	2703	10	143
3	Fa	174.6	2864	11	48
3	Fa#	185.0	3035	11	219
3	Sol	196.0	3215	12	143
3	Sol#	207.7	3407	13	79
3	La	220.0	3609	14	25
3	La#	233.1	3824	14	240
3	Si	246.9	4050	15	210

Valori delle note

Ottava	Nome della nota	Frequenza in hertz	Regolaz. di freq. del SID	Byte alto della regol. di freq. del SID	Byte basso della regol. di freq. del SID
4	Do	261.6	4291	16	195
4	Do#	277.2	4547	17	195
4	Re	293.7	4818	18	210
4	Re#	311.1	5103	19	239
4	Mi	329.6	5407	21	31
4	Fa	349.2	5728	22	96
4	Fa#	370.0	6070	23	182
4	Sol	392.0	6431	25	31
4	Sol#	415.3	6813	26	157
4	La	440.0	7218	28	50
4	La#	466.2	7648	29	224
4	Si	493.9	8102	31	166
5	Do	523.3	8584	33	136
5	Do#	554.4	9095	35	135
5	Re	587.3	9634	37	162
5	Re#	622.3	10208	39	224
5	Mi	659.3	10815	42	63
5	Fa	698.5	11458	44	194
5	Fa#	740.0	12139	47	107
5	Sol	784.0	12861	50	61
5	Sol#	830.6	13625	53	57
5	La	880.0	14436	56	100
5	La#	932.3	15294	59	190
5	Si	987.8	16204	63	76
6	Do	1046.5	17167	67	15
6	Do#	1108.7	18188	71	12
6	Re	1174.7	19270	75	70
6	Re#	1244.5	20415	79	191
6	Mi	1318.5	21629	84	125
6	Fa	1396.9	22915	89	131
6	Fa#	1480.0	24278	94	214
6	Sol	1568.0	25722	100	122
6	Sol#	1661.2	27251	106	115
6	La	1760.0	28872	112	200
6	La#	1864.7	30589	119	125
6	Si	1975.5	32407	126	151
7	Do	2093.0	34334	134	30
7	Do#	2217.5	36377	142	25
7	Re	2349.3	38539	150	139
7	Re#	2489.0	40831	159	127
7	Mi	2637.0	43258	168	250
7	Fa	2793.8	45831	179	7
7	Fa#	2960.0	48557	189	173
7	Sol	3136.0	51444	200	244
7	Sol#	3322.4	54502	212	230
7	La	3520.0	57743	225	143
7	La#	3729.3	61177	238	249
7	Si	3951.1	64815	253	47

AND e OR

AND e OR sono le operazioni logiche che il vostro Commodore 64 utilizza per manipolare i bit e per controllare la verità di espressioni complesse. Cercherò di darvi rapidamente un'idea di come funzionano. Prima di tutto, qualche convenzione:

- Quando il computer cerca di decidere se un numero è vero o falso, tutti i numeri diversi da zero sono considerati veri.
- Quando il calcolatore effettua un confronto, e decide che il confronto stesso è vero, assegna il valore −1. Quando il confronto da' risultato falso, viene assegnato il valore 0.

Ecco un breve programma che illustra queste due convenzioni al lavoro:

```
10  IF 8 THEN PRINT "8 E' VERO"
20  IF 0 THEN PRINT "0 E' VERO"; GOTO 40
30  PRINT "0 E' FALSO"
40  PRINT (9 = 8)
50  PRINT (9 = 9)
```

Eseguendo il programma, si ottengono questi risultati:

```
8  E' VERO
0  E' FALSO
0
-1
```

Il Commodore 64 effettua le operazioni di AND e OR su numeri compresi nell'intervallo da −32768 a +32767. I numeri vengono prima privati

delle eventuali parti frazionarie, poi vengono trasformati nella forma binaria a 16 bit. Ecco qualche esempio:

Valore originario	Dopo eliminazione della parte frazionaria	Binario a 16 bit
−1	−1	1111 1111 1111 1111
254.75	254	0000 0000 1111 1110
513	513	0000 0010 0000 0001
0	0	0000 0000 0000 0000
15.4	15	0000 0000 0000 1111

Notate che inserito degli spazi nei valori binari a 16 bit solamente per renderne più facile la lettura.
Quando si effettua l'AND di due numeri, questi dapprima vengono messi in questa forma binaria a 16 bit, privi della parte frazionaria; poi si effettua l'AND dei bit corrispondenti, sulla base delle seguenti regole:

```
    0        0        1        1
AND 0    AND 1    AND 0    AND 1
-----    -----    -----    -----
    0        0        0        1
```

I risultati poi sono trasformati di nuovo nella forma decimale. Ecco qualche esempio di applicazione dell'AND:

```
                         -1       decimale
                   AND    0       decimale
------------------------------------------
     1111  1111  1111  1111       binario
AND  0000  0000  0000  0000       binario
------------------------------------------
     0000  0000  0000  0000       binario
------------------------------------------
                          0       decimale

                        255       decimale
                   AND   15       decimale
------------------------------------------
     0000  0000  1111  1111       binario
AND  0000  0000  0000  1111       binario
------------------------------------------
     0000  0000  0000  1111       binario
                         15       decimale
```

Nella programmazione per la grafica e i suoni sul Commodore 64, l'AND viene usato spesso per disattivare taluni bit in un registro. Per esempio,

se si vogliono disattivare i bit 4, 5, 6 e 7 di un registro, si effettua l'AND fra il valore del registro e il numero 15. Date un'occhiata all'ultimo esempio, e capirete perché.
Quando si effettua l'OR fra due numeri, questi vengono dapprima portati nella forma, ormai familiare, di numero binario a 16 bit, privati della parte frazionaria. Si effettua l'OR dei bit corrispondenti sulla base delle regole seguenti:

	0		0		1		1
OR	0	OR	1	OR	0	OR	1
	0		1		1		1

Il risultato poi viene nuovamente riportato alla forma decimale. Ecco alcuni esempi di applicazione dell'OR:

				−1	decimale
			OR	0	decimale
	1111	1111	1111	1111	binario
OR	0000	0000	0000	0000	binario
	1111	1111	1111	1111	binario
				−1	decimale

				537	decimale
			OR	131	decimale
	0000	0010	0001	1001	binario
OR	0000	0000	1000	0011	binario
	0000	0010	1001	1011	binario
				67	decimale

Nella programmazione per la grafica e i suoni sul Commodore 64, l'OR viene usato spesso per attivare taluni bit in un registro. Per esempio, se si vogliono attivare i bit 0, 1 e 7 di un registro, si effettua l'OR fra il valore del registro e il numero 131. Date un'occhiata all'ultimo esempio e capirete perché.
Questo è solo un breve sguardo a AND e OR: in effetti si tratta di funzioni davvero notevoli. Il vostro Commodore 64, al livello più profondo, passa la maggior parte del proprio tempo a effettuare operazioni di AND e OR vari milioni di volte al secondo.

Le capacità di elaborazione grafica e sonora sono le caratteristiche più interessanti del Commodore 64, macchina domestica dalle prestazioni raffinate; ma le tecniche di programmazione per sfruttare queste capacità richiedono uno studio autonomo rispetto a quello del Basic per la (normale) elaborazione di informazioni, che peraltro sono presupposte.

Questo volume è una guida alle capacità grafiche e sonore del Commodore 64 che prende per mano il lettore, conducendolo dai primi rudimenti fino a una buona conoscenza delle funzioni degli integrati VIC-II e SID (preposti rispettivamente alla grafica e alla generazione dei suoni) e della loro programmazione. Il testo è corredato da molti programmi esemplificativi completi, che si prestano bene alla sperimentazione da parte del lettore. Ogni capitolo è completato da un test di autovalutazione e da esercizi di programmazione, per i quali l'autore fornisce anche possibili soluzioni.

franco muzzio & c. editore

ISBN 88-7021-273-4

L. 22.000 (21.568)